Jül-4330

Jülich Supercomputing Center (JSC)

Gradienten basiertes Lastbalance-Verfahren für teilchenbasierte Simulation

Jens Freche

Mitglied der Helmholtz-Gemeinschaft

JÜLICH
FORSCHUNGSZENTRUM

Gradienten basiertes Lastbalance-Verfahren für teilchenbasierte Simulation

Jens Freche

Berichte des Forschungszentrums Jülich; 4330
ISSN 0944-2952
Jülich Supercomputing Center (JSC)
Jül-4330

Vollständig frei verfügbar im Internet auf dem Jülicher Open Access Server (JUWEL)
unter http://www.fz-juelich.de/zb/juwel

Zu beziehen durch: Forschungszentrum Jülich GmbH · Zentralbibliothek, Verlag
D-52425 Jülich · Bundesrepublik Deutschland
☎ 02461 61-5220 · Telefax: 02461 61-6103 · e-mail: zb-publikation@fz-juelich.de

Zusammenfassung der Arbeit

Das hochskalierte Programm MP2C (Massively Parallel Multi-Particle Collision) wird zur Zeit im Simulation Laboratory Molecular Systems des JSC entwickelt. Das Programm simuliert Teilchen und deren Wechselwirkungen auf der Basis von Molekulardynamik und der Hydrodynamik. Das mit MPI parallelisierte Programm basiert auf einer Gebietszerlegung, so dass das Simulationsgebiet in geometrische Bereiche gleichen Volumens unterteilt wird und jeder Prozessor die Teilchen eines Gebietes bearbeitet. Durch die Bewegung der Teilchen kommt es im Laufe der Simulation zu einer ungleichen Verteilung der Teilchen auf die einzelnen Gebiete. Zur Optimierung der Performance muss ein Lastbalance-Verfahren dafür Sorge tragen, dass dieses Ungleichgewicht beseitigt wird und jeder Prozessor in etwa gleich ausgelastet ist.

Diese Arbeit beschäftigt sich mit einem Loadbalancing-Verfahren, welches auf Gradienten basiert und die Grundlage für einen weiteren Lastbalance-Ansatz im Programm MP2C bietet. In diesem Verfahren werden Gebietsgrenzen dadurch verändert, in dem man Gitterpunkte verschiebt. Dabei werden die Gitterpunkte von Gebieten mit viel Arbeit angezogen und von Gebieten mit wenig Arbeit abgestoßen. In dem Zusammenhang werden zudem effiziente und allgemeine Methoden erarbeitet, die testen, ob Teilchen innerhalb eines Gebietes liegen, welches durch seine Eckpunkte eindeutig festgelegt ist. Zuletzt werden die Ergebnisse der Implementierungen analysiert und aufgezeigt, inwiefern das erarbeitete Loadbalancing-Verfahren praktikabel für MP2C umzusetzen ist.

Summary

The highly scalable program Massively Parallel Multi-Particle Collision (MP2C) is currently developed in the Simulation Laboratory Molecular Systems at Juelich Supercomputing Centre. MP2C simulates particles and their interactions based on molecular dynamics and hydrodynamics. The MPI parallelized program is based on a domain decomposition, i. e. the system in divided into equal-sized geometrical domains and each processor handles the particles of one domain. Due to the movement of the particles, the distribution of particles among the domains can become imbalanced during the simulation. To optimize the performance, a load balancing method is used which ensures that the workload is equally distributed among the processes.

This thesis introduces a load balancing method which bases on gradients. In this method, grid points are moved to change the volume of the domains in a way that domains with lots of particles attract grid points whereas domains with only a few particles repel them. In this regard, an efficient algorithm to determine the correct domain containing the particle is developed. Finally, the results are analyzed and the practicability for MP2C is discussed.

Inhaltsverzeichnis

Kapitel 1

Ausgangspunkt und Motivation

Im ersten Teil dieser Arbeit wird das Thema „Gradienten basiertes Lastbalance-Verfahren für teilchenbasierte Simulation“motiviert und erläutert, weshalb es sinnvoll ist sich mit dem Thema Lastbalance zu beschäftigen. Außerdem wird der Zusammenhang zwischen dieser Arbeit und dem Programmpaket MP2C hergestellt. Das Lastbalance-Verfahren mittels Gradienten wird in den folgenden Kapiteln herausgearbeitet und von bereits vorhandenen Loadbalancing-Strategien abgegrenzt und mit diesen verglichen.

1.1 Grundlagen und Ziele des Programmpaketes MP2C

Die Grundlage für das Programmpaket MP2C bilden die Molekulardynamik (MD) und die Multiparticle Collision Dynamics (MPC). In MP2C werden MD und MPC gekoppelt. In dieser Arbeit wird jedoch ausschließlich auf MD eingegangen und MPC nicht weiter berücksichtigt. In der MD werden Teilchensysteme und deren Wechselwirkungen untersucht und mit Hilfe von Computern simuliert. In diesen Simulationen wird die eigentlich kontinuierliche Zeitskala diskretisiert. An diesen diskreten Zeitpunkten werden die Wechselwirkungen im System berechnet. Durch die Wechselwirkungen der

Teilchen sind deren Orientierung und Geschwindigkeit eindeutig bestimmt. Eine detailliertere Beschreibung der in der MD berücksichtigten Wechselwirkungen ist Kapitel 2.1 zu entnehmen. Die Unterteilung der Wechselwirkungen in kurzreichweitig und langreichweitig wird dort genauer erläutert. Insbesondere die Komplexität der Simulation und die Performance werden grundlegend von der Frage, ob es sich um kurzreichweitige oder langreichweitige Wechselwirkungen handelt, beeinflusst. Das Programmpaket MP2C simuliert ausschließlich kurzreichweitige Wechselwirkungen. Bei dieser Art der Wechselwirkung interagieren Teilchen nur mit anderen Teilchen, welche im näheren Umfeld liegen. Diese Eigenschaft bietet die Möglichkeit einer effizienten Parallelisierung, weil Interaktionen in unterschiedlichen Gebieten des Systems unabhängig voneinander ablaufen können. Da durch eine Parallelisierung die Ausführungszeit des Programms deutlich verringert werden kann und sie durch die beschriebene Unabhängigkeit möglich ist, wird in dieser Arbeit ebenfalls auf die Grundlagen der Parallelisierung eingegangen und beschrieben, wie das Lastbalance-Verfahren parallel zu implementieren ist. Die Parallelisierung des Programms soll zusätzlich auf massiv-parallelen Hochleistungsrechnern ausgeführt werden können und für eine große Anzahl an Prozessoren gut skalieren. In diesem Zusammenhang ist es von großer Bedeutung, dass eine sehr gute und gleichmäßige Arbeitsaufteilung auf die Prozessoren sichergestellt wird, um dadurch ein überproportionales Ansteigen an Kommunikation und längerer Wartezeiten zu vermeiden. Diese gleichmäßige Arbeitsaufteilung kann mit dem Lastbalance-Verfahren dieser Arbeit erreicht werden.

Das Programmpaket MP2C simuliert Teilchensysteme auf parallelen Rechnern. Da die Teilchen in diesen Systemen im Allgemeinen inhomogen verteilt sind und daher auch eine Lastinbalance vorliegt, ist man daran interessiert dieses Ungleichgewicht auszugleichen. Diese Arbeit stellt die Grundlage für ein Verfahren dar, um diese Lastinbalance auszugleichen und im Programmpaket MP2C anzuwenden können.

Kapitel 2

Allgemeine Grundlagen

2.1 Molekulardynamik

MD beschäftigt sich mit dem Lösen der klassischen Bewegungsgleichungen für ein System von N Teilchen. Unabhängig von der heutigen Molekulardynamik formulierte Pierre-Simon de Laplace Ende des 18. Jahrhunderts die folgende Aussage: „Eine Intelligenz, die zu jedem Zeitpunkt die Kräfte vorhersehen könnte, die die Natur in Bewegung versetzen und die Position aller Teile, aus der sie aufgebaut ist, [...] nichts wäre unvorherbestimmt und sowohl die Zukunft, als auch die Vergangenheit wären seinen Augen zugänglich“ [17]. Diese bereits vor ca. 200 Jahren gemachte Aussage passt gut zu der Grundidee der MD. Die Intelligenz von der Laplace spricht, könnte in vielen Punkten eben genau durch die MD ersetzt werden. Es wird versucht die Natur möglichst detailgetreu abzubilden und zu simulieren. Man kennt die mikroskopischen Zustände der Natur aber zu schlecht, als dass ganz eine genaue Vorhersage möglich wäre. Im Rechner sind Positionen und Geschwindigkeit eindeutig gegeben, aber die genaue Beschreibung der zeitliche Entwicklung ist limitiert. Dies ist durch Rundungsfehler und die zeitlichen Diskretisierung zu erklären. Folglich kann man die Natur recht gut nachbilden, aber nicht exakt vorhersagen. Die Wechselwirkung zwischen den Teilchen wird über ein Potential bzw. die Gesamtenergie des Systems definiert. Eines der gängigsten Potentiale ist

das „Lennard-Jonas Potential“. Dieses Potential wird so formuliert:

$$u(r) = 4\left(\left(\frac{1}{r}\right)^{12} - \left(\frac{1}{r}\right)^{6}\right) \tag{2.1}$$

Die Kraft, welche in einem System wirkt und für die Bewegung der Teilchen verantwortlich ist, erhält man über die negative Ableitung des Potentials nach dem Ort. Die Gesamtenergie H eines Hamilton-Systems ist die Summe aus der kinetischen Energie und dem Potential.

$$\begin{aligned} H &= K + U \\ &= \frac{1}{2}\sum_{i_1=0}^{N}\frac{p_i^2}{m_i} + \sum_{i,j>i}^{N} u(r_{ij}) \end{aligned}$$

Hier beschreibt die Variable r_{ij} den Abstandes zwischen den Teilchen i und j. Der Impuls p_i des i-ten Teilchens ist definiert über seine Masse und Geschwindigkeit: $p_i = m_i v_i$
Für die Geschwindigkeit v_i und die Kraft F_i ergeben sich folglich:

$$\dot{r}_i = \frac{\partial H}{\partial p_i} = \frac{p_i}{m_i} = v_i$$

$$\dot{p}_i = -\frac{\partial H}{\partial r_i} = -\frac{\partial U}{\partial r_i} = F_i$$

Zur Bestimmung des Ortes eines Teilchens zum Zeitpunkt $t + \delta t$, kann die Taylorentwicklung verwendet werden:

$$r\left(t + \delta t\right) \approx r\left(t\right) + v\left(t\right)\delta t + \frac{1}{2}\frac{F\left(t\right)}{m}\delta t^2$$

Zudem wird in der MD zwischen kurzreichweitigen und langreichweiten Wechselwirkungen unterschieden. Diese Unterscheidung wird durch das Potential bestimmt. Liegt ein Potential der Form $\frac{1}{r^n}$ vor mit $n \leq d$, wobei d die Dimension des Systems bestimmt, so liegt eine langreichweitige Wechselwirkung vor.

Falls $n \geq d$ gilt, so liegt, wie für das „Lennard-Jones Potential“ in 2.1, eine kurzreichweitige Wechselwirkung vor. Bei der langreichweitigen Wechselwirkung wird davon ausgegangen, dass alle Teilchen miteinander interagieren. Dies führt zu einer Komplexität $O(N^2)$, wobei N die Anzahl der Teilchen des gesamten Systems ist. Bei kurzreichweitigen Wechselwirkungen werden nur die Nachbarteilchen berücksichtigt, die innerhalb eines bestimmten Cut-Off Radius liegen.
Häufig werden in der MD sogenannte Linked-Cell-Listen verwendet. Das Ziel dieser Listen besteht darin, die Komplexität zu verringern. Außerdem sollen nur lokale Operationen benutzt werden. Das gesamte System wird in quaderförmige Zellen unterteilt. Für jede Zelle werden im zweidimensionalen eindeutig 8 Nachbarzellen und im dreidimensionalen 26 Nachbarzellen identifiziert. Innerhalb einer Zelle werden die Teilchen in einer bestimmten Reihenfolge in Listen abgespeichert, in den beschriebenen Linked-Cell-Listen. Diese Listen werden zellenweise durchlaufen. Es wird somit eine große Kraftschleife in viele kleinere Kraftschleifen aufgespaltet. Der Speicherbedarf wächst auf Grund der einzelnen Listen nur mit $O(N)$. Außerdem werden keine globalen Operationen für den Listenaufbau benötigt. [17]
Das allgemeine Linked-Cell-Verfahren kann jedoch nicht ohne Weiteres mit dem in dieser Arbeit erarbeiteten Loadbalancing-Verfahren gekoppelt werden, da die Listen rechtwinklige Zellen voraussetzen und die Loadbalancing-Schritte im Wesentlichen mit nichtkubischen Gebieten arbeiten.

2.2 Vorteile und Probleme der Parallelisierung

Wie bereits im vorangegangenen Kapitel erklärt, ist ein Ziel von MP2C eine möglichst gute Skalierbarkeit für eine hohe Anzahl von Prozessoren zu erreichen.
Um ein Programm möglichst gut parallelisieren zu können, müssen verschiedene Faktoren berücksichtigt werden. Ein erstes wichtiges Kriterium bildet dabei der Speedup. Allgemein ist der Speedup ein Maß für die Bewertung der Verarbeitung eines einzelnen Auftrages. Bezogen auf die Parallelisierung

wird der Speedup S über die sequentielle Laufzeit t_{seq} und parallele Laufzeit t_{para} definiert:

$$S = \frac{t_{seq}}{t_{para}}$$

Für eine erste genauere Betrachtung des Speedups wird seit 1967 das Gesetz von Amdahl verwendet. Zur Definition dieses Gesetzes unterteilte der amerikanisch-norwegische Computerarchitekt Gene Myron Amdahl jedes Programm in zwei Anteile. Zum Einen in den Teil der parallelisiert werden kann und zum Anderen in den Programmteil der inhärent sequentiell ist und somit nicht zu parallelisieren ist. Unter dieser Annahme kann man den Speedup in Abhängigkeit von der Anzahl der verwendeten Prozessoren folgendermaßen definieren:

$$S(p) = \frac{t_{seq}}{t_{para}} = \frac{t_{seq}}{f \cdot t_{seq} + (1-f) \cdot \frac{t_{seq}}{p}}$$

In dieser Formel steht p für die Anzahl der Prozessoren, t_{seq} für die Laufzeit der sequentiellen Abarbeitung des gesamten Programms und f beschreibt den Anteil der Laufzeit, welcher für Teile verbraucht wird, die nicht parallel ablaufen können. Würde man annehmen, dass der nicht parallelisierbare Anteil f verschwindend gering ist, so ergibt sich für den Speedup der folgende Ausdruck:

$$\lim_{f \to 0} S(p) = p$$

Für eine effiziente Parallelverarbeitung ist es demnach erstrebenswert, einen Geschwindigkeitsgewinn bzw. Speedup zu erzielen, der näherungsweise der Prozessorzahl entspricht.
Bildet man im nächsten Schritt das Verhältnis aus dem Speedup und der eingesetzen Prozessorzahl, so erhält man die Effizienz E der Parallelisierung. Sie gibt an, welcher Anteil der Prozessorleistung nutzbar ist.

$$E(p) = \frac{S(p)}{p} = \frac{t_{seq}}{p \cdot t_{para}}$$

Erstrebenswert ist eine Effizienz, die für möglichst hohe Prozessorzahlen p einen Wert nahe bei eins liefert. In der Betrachtung von Amdahl wird davon ausgegangen, dass die Problemgröße unverändert bleibt. Die Abschätzung

für den theoretischen Speedup ist dabei sehr pessimistisch. Günstigere Voraussetzungen ergeben sich, falls man für die gleiche Zeit die Problemgröße wachsen lässt, da dann der parallele Anteil höher ist. Diese Überlegung wurde im Gustafson'schen Gesetz [10]zusammen gefasst.
Die zwei wichtigsten Unterschiede zu der Vorgehensweise von Amdahl sind:

- Die Problemgröße wird so an die Prozessorzahl p angepasst, dass die parallele Ausführungszeit konstant bleibt.
- Der absolute sequentielle Anteil des Programms ft_{seq} ist konstant. D.h. unabhängig von der Problemgröße gibt es einen Teil des Programms, welcher nicht parallelisierbar ist.

Daraus ergibt sich, dass die sequentielle Verarbeitungszeit mit steigender Prozessorzahl wächst

$$t_{seq} = ft_{seq} + (1 - f)t_{seq}$$

wohin gegen die parallele Ausführungszeit konstant bleib:

$$t_{para} = ft_{seq} + (1 - f)\frac{t_{seq}}{p}$$

Der Speedup nach Gustafson lautet:

$$\begin{aligned} S(p) &= \frac{ft_{seq} + p\frac{(1-f)t_{seq}}{p}}{t_{para}} \\ &= \frac{pt_{para} - (p-1)ft_{seq}}{t_{para}} \\ &= p - \frac{(p-1)ft_{seq}}{t_{para}} \\ &= p(1 - \frac{ft_{seq}}{t_{para}}) + \frac{ft_{seq}}{t_{para}} \end{aligned}$$

Unter der oben gemachten Annahme, dass die Prozessorzahl proportional zu der Problemgröße wächst, sind ft_{seq} und t_{para} und somit auch $\frac{ft_{seq}}{t_p}$ konstant. Der Quotient gibt dabei den seriellen Anteil der parallelen Ausführungszeit an. Damit ergibt sich, dass der Speedup linear in p ist. Die Steigung des

Speedups beträgt $1 - \frac{ft_{seq}}{t_{para}}$ und wird faktisch immer kleiner eins sein.
Die am Anfang dieses Kapitels geforderte Skalierbarkeit, die das Zusammenwirken von Software und Hardware beschreibt, wird maßgeblich vom Speedup des Programms bestimmt.[10]

Die Parallelisierung in der MD und somit auch im Programmpaket MP2C ist wesentlich davon abhängig, wie man die Arbeit im gegebenen System aufteilt. Wenn man beispielsweise Polymere oder Blutkörperchen simuliert, dann ist nicht davon auszugehen, dass die zu simulierenden Teilchen homogen im Raum verteilt sind. So sind lineare Polymere in Ketten angeordnet, die das System nicht gleichmäßig ausfüllen. Man kann davon ausgehen, dass die Teilchen sehr inhomogen über das System verteilt sind. Dabei können hohe Konzentrationen von Teilchen an bestimmten Stellen auftreten, andere Gebiete hingegen können gar keine Teilchen beinhalten.
Allgemein unterscheidet man bei der parallelen Aufteilung der Arbeit zwischen Domain Decomposition, Force Decomposition und Functional Decomposition. Bei der Functional Decomposition werden unterschiedliche Programmteile, Unterprogramme und Module auf die verschiedenen Prozessoren verteilt und parallel bearbeitet. Hier kann man verschiedene Rechnerarchitekturen für unterschiedliche Programmteile einsetzen. In der MD wird jedoch überwiegend der Ansatz der Domain Decompsition verwendet.
Diese Zerlegung beschreibt allgemein die Aufteilung eines Datengebiets in Bereiche gleicher Größe. Anders als bei der Functional Decomposition werden alle Gebiete mit dem selben Programmteil bearbeitet. Die einzelnen Gebiete werden auf die unterschiedlichen Prozesse aufgeteilt. Jeder Prozess berechnet dabei die Geschwindigkeiten und Interaktionen in seinem Gebiet. Informationen, die auch für die Nachbargebiete von Bedeutung sind, müssen zusätzlich kommuniziert werden.
Bei inhomogenen Teilchensytemen kann es vorkommen, dass einige Prozessoren besonders viele Teilchen besitzen, während andere Prozessoren sehr wenige oder sogar gar keine Teilchen bearbeiten. Ein solches Ungleichgewicht in der Arbeitsverteilung hat zur Folge, dass die verwendeten Prozessoren nicht optimal ausgenutzt werden, da Prozessoren mit wenig Arbeitslast auf solche

mit viel Arbeitslast warten müssen. Es gilt also eine Methode zu finden, die diese ungleiche Arbeit auf die einzelnen Prozessoren ausgleicht oder ausbalanciert.
Außerdem kann es auch bei der parallelen Ausgabe zu Problemen kommen. Falls die Ausgabe beispielsweise nur sequentiell abläuft, kann es passieren, dass der eigentlich gute Speedup bzw. die gute Skalierbarkeit verloren geht. Abhilfe können dabei der parallele I/O von MPI oder die Benutzung von SIONlib [5] leisten. Auf die Problematik der parallelen Ein-/Ausgabe wird in dieser Arbeit jedoch nicht genauer eingegangen.

2.3 Lösungsmöglichkeiten

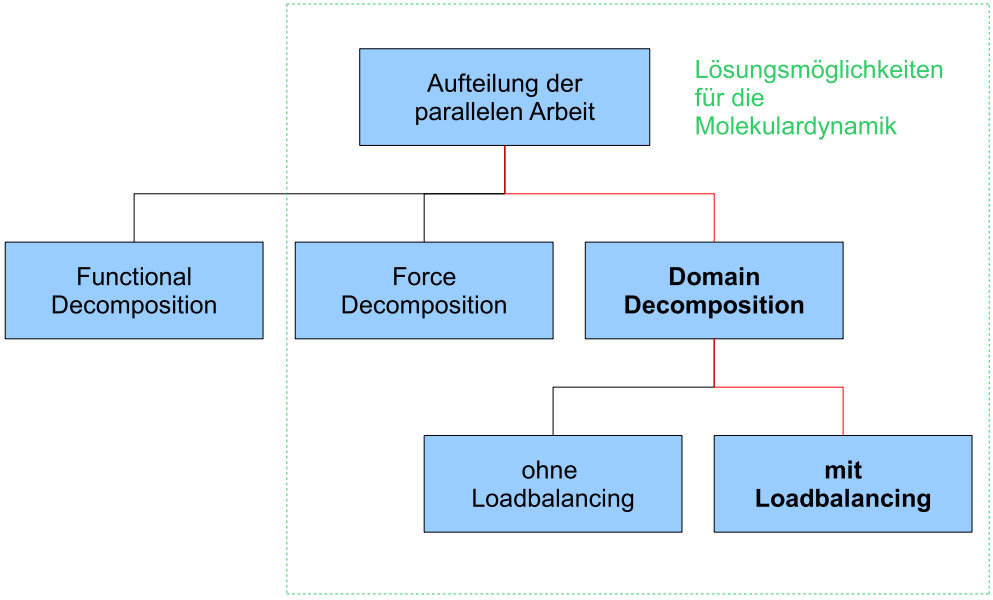


Abbildung 2.1: Möglichkeiten der Parallelisierung

Die Graphik 2.1 zeigt die Möglichkeiten, wie Arbeit aufgeteilt und parallelisiert werden kann. Wie bereits im Kapitel 2.2 beschrieben, ist die Functional Decomposition im Allgemeinen kein probates Mittel für die Molekulardynamik, da die Kräfteberechnung der rechenintensivste Teil eines MD-Programms ist und dieser nicht mit der Functional Decomposition auf ver-

schiedene Prozessoren aufgeteilt werden kann. Zur Lösung der molekulardynamischen Probleme eignen sich die Domain Decomposition und die Force Decomposition. Die Informationen zur Berechnung der Wechselwirkungen der Teilchen werden bei der Force Decomposition als Matrix notiert. Diese Matrix wird gleichmäßig auf die unterschiedlichen Prozessoren verteilt, so dass die zu verrichtende Arbeit möglichst ausgeglichen ist. Der eindeutigen und gut handhabbaren Matrix steht aber ein hoher Kommunikationsaufwand gegenüber, um die Kräfte bzw. Matrixeinträge auf die anderen Prozesse zu verteilen.
Bei der Domain Decomposition, berechnet jeder Prozessor die Geschwindigkeiten und Wechselwirkungen für die Teilchen, die in seinem Gebiet liegen. Führt man diese Aufteilung durch, ohne dabei die Gebietsgrenze zu verändern, so ist das Verfahren, wie in Kapitel 2.2 beschrieben, für inhomogene Teilchensysteme sehr ineffizient. Wenn man die Veränderung der Gebietsgrenzen zusätzlich erlaubt, um die Arbeit auf den einzelnen Prozessoren auszugleichen, spricht man von Domain Decomposition mit Loadbalancing (s. Graphik 2.1, roter Pfad).
Im Hauptteil dieser Arbeit (Kapitel 3) wird ein Loadbalancing-Verfahren erarbeitet und von bereits existierenden Verfahren abgegrenzt. Diese bereits vorhandenen Loadbalancing-Strategien werden unter 2.5 kurz vorgestellt. Darüber hinaus ist es von Vorteil, wenn Loadbalancing-Verfahren weitesgehend nur auf lokale Informationen und Informationen von unmittelbar benachbarten Prozessoren zugreifen, da dann der Kommunikationsaufwand klein gehalten werden kann und nur Kommunikation mit direkten Nachbarn nötig ist. Das Gitter für die angepasste Gebietseinteilung bei inhomogenen Systemen unterscheidet sich in der Regel sehr deutlich von der äquidistanten Unterteilung eines Systems. [6]

2.4 Parallelisierte Molekulardynamik

In der Molekulardynamik kann es zu sehr umfangreichen numerischen Rechnungen kommen. Da diese numerischen Rechnungen für realistische Systeme

nicht von einem Rechner bewältigt werden können, verteilt man diese Arbeit auf mehrere Rechner, welche die Teilaufgaben parallel abarbeiten. Unter der Voraussetzung, dass das ursprüngliche Problem parallelisierbar ist, müssen für eine gute Parallelisierung adäquate parallele Algorithmen verwendet werden. Die Implementierung muss zudem effizient geschehen, so dass beispielsweise keine Bottlenecks entstehen. In Kapitel 2.3 wurde bereits darauf eingegangen, wie Arbeit allgemein parallel aufgeteilt werden kann. In der Molekulardynamik können die Strategien zur Parallelisierung noch genauer untergliedert werden. Neben den bereits bekannten Aufteilungen der Force-Decomposition, der Domain-Decomposition und der Functional Decomposition sind in der MD außerdem die folgenden Strategien denkbar und/oder gebräuchlich [17]:

- Cloning / Farming
- Master-Slave
- Particles-Decomposition

Cloning / Farming

Bei der Strategie des Cloning wird das gesamte System auf jedem Prozessor bekannt gemacht. D.h. jeder Prozessor besitzt alle N Teilchen und führt die gesamte Simulation durch. Dies geschieht auf jedem Prozessor mit unterschiedlichen Startbedingungen. Im nächsten Schritt wird in jedem Durchlauf und über alle Prozessoren der Mittelwert über die Kräfte gebildet. Bei dieser Art der Parallelisierung erreicht man normalerweise einen idealen Speedup, da es geringen Overhead durch Kommunikation gibt. Faktisch kann das Cloning jedoch nur für kleine Systeme und kleine Zeitskalen benutzt werden. Für größere Systeme ist er nicht praktikabel, da alle Teilchen auf jedem Prozessor gerechnet werden und die Ausführungszeit somit extrem hoch wäre. Diese Vorgehensweise wird in der statistischen Mechanik verwendet, wird aber in dieser Arbeit und in dem Programm MP2C nicht weiter berücksichtigt.

Master-Slave

Es wird ein Masterprozessor bestimmt, der das gesamte Loadbalancing kontrolliert. Dabei verteilt er die Arbeit und Daten auf alle Prozessoren. Die Aufgaben können somit gleichzeitig abgearbeitet werden. Bevor dieser parallele Ablauf stattfinden kann, muss der Masterknoten jedoch alle Daten vorhalten und verteilen, so dass ein großer Kommunikationaufwand zu Beginn der Simulation entsteht. Dieser Bottleneck in der Kommunikation und der große Speicherbedarf auf dem Masterknoten führen dazu, dass die Master-Slave-Parallelisierung in der MD nicht weit verbreitet ist.

Particle-Decomposition

In dieser Aufteilung werden N Teilchen auf P Prozessoren verteilt: $N_P = \frac{N}{P}$. Die Anzahl der Teilchen N_P pro Prozessor bleibt hier während der gesamten Simulation konstant. Jeder Prozessor aktualisiert den Ort und die Geschwindigkeit „seiner“ N_P Teilchen. Da zur Berechnung der neuen Orte und Geschwindigkeiten in der Regel auch die Informationen von Teilchen anderer Prozessoren benötigt werden, muss zusätzlich ein Algorithmus zum Austausch dieser Daten gefunden werden. Häufig wird für dieses Replizieren der Daten der sogenannte systolische Ring verwendet. Bei dem Verteilen der Daten mittels des systolischen Rings werden schnelle Algorithmen verwendet, die auf einer kreisförmigen Anordnung der Prozessoren basieren und sicherstellen, dass jeder Prozessor nach dem Algorithmus die Dateien aller anderen Prozessoren kennt. Da die Kommunikation zum Verschieben der Daten für MP2C zu aufwendig ist und somit nicht angewendet wird, wird der Ansatz der Particle-Decomposition an dieser Stelle ebenfalls nicht weiter verfolgt.
Die im Kapitel 3 beschriebenen Verfahren zur Gebietszerlegung basieren ausschließlich auf der in 2.3 erläuterten Domain-Decomposition, also der geometrischen Aufteilung des Systems auf die unterschiedlichen Prozessoren.

2.5 Weitere Loadbalancing-Ansätze

In diesem Abschnitt werden zwei weitere Loadbalancing-Strategien beschrieben. Das erste Verfahren ist ein Loadbalancing-Verfahren, welches von Herrn Bücker am JSC (Jülich Supercomputing Centre) [2] im Rahmen der Masterarbeit entwickelt wurde. Es basiert auf der FMM (Fast Multipole Method) und führt für die darin enthaltende raumfüllende Kurve, sowie für deren Blöcke ein Loadbalancing durch.

2.5.1 Verteilung der Wechselwirkungen

Dem Loadbalancing-Verfahren von Bücker liegt die FMM zu Grunde. Ein wesentliches Merkmal der FMM ist, dass die Wechselwirkung von Teilchen mit anderen Teilchen, die weiter entfernt liegen, abstrahiert werden.

Diese Abstraktion liegt darin, dass in der langreichweitigen Wechselwirkung die Wechselwirkung nicht zu allen Teilchen des Systems einzeln berechnet wird. Weiter entfernt liegende Teilchen werden in bestimmte Gruppen zusammengefasst. D.h. Teilchen, die entfernt liegen bezogen, auf das Teilchen zu dem die Wechselwirkung berechnet wird, aber untereinander näher zusammen liegen werden zu einem Multipol zusammen gefasst. Zwischen diesem Multipol und dem Teilchen, zu dem die Wechselwirkung bestimmt werden soll wird nur einmal die Wechselwirkung berechnet. Dafür verwendet man den Massenschwerpunkt aller Teilchen des Multipols und gewichtet ihn mit der Anzahl der im Multipol enthaltenen Teilchen. Außerdem sind Teilchen in diesem Verfahren in Boxen einsortiert. Für die acht Nachbarboxen in zwei Dimensionen bzw. 26 Nachbarboxen in drei Dimensionen wird die Wechselwirkung direkt zu jedem Teilchen berechnet. Die Boxen, in denen die Teilchen einsortiert sind, liegen auf einer Raumfüllenden Kurve. Diese raumfüllenden Kurven haben die Eigenschaft, dass mit allen direkten Nachbarboxen möglichst unkompliziert kommuniziert werden kann. Die verwendeten raumfüllenden Kurven sind dabei die Hilbert-Kurve [9], die Gosper-Kurve und die Z-Kurve [12]. Auf die genaue Nummerierung und Zusammensetzung wird in

dieser Arbeit nicht näher eingegangen.
Jeder Prozessor bekommt im nächsten Schritt eine bestimmte Anzahl an Boxen zu geordnet, wobei darauf geachtet wird, dass die Anzahl der Teilchen auf jedem Prozessor möglichst gleich ist. Da aber die Teilchen in Boxen einsortiert sind, und diese Boxen nur vollständig auf einem Prozessor liegen können, kann es zu einer unterschiedlichen Anzahl an Teilchen auf jedem Prozessor kommen. An dieser Stelle greift nun das Loadbalancing-Verfahren von Bücker ein. Dabei wird der Mittelwert der Teilchen pro Prozessor bestimmt und ein Konfidenzintervall um diesen Mittelwert gelegt. Die Größe dieses Intervalls wird vorab festgelegt. Hat ein Prozessor so wenig Teilchen, dass die Anzahl der Teilchen unterhalb des Konfidenzintervalls liegt, so bekommt er von Prozessoren, deren Teilchenanzahl oberhalb des Konfidenzintervalls liegt, eine Box übergeben. Diese Übergabe von Boxen geschieht solange bis die Teilchenanzahl aller Prozessoren im Konfidenzintervall liegen. Mögliche Probleme, die dabei auftreten können sind der Arbeit von Bücker zu entnehmen.
Das Programm MP2C fordert räumlich zusammenhängende Gebiete von benachbarten Prozessoren. Dieser Zusammenhang ist mit dem Ansatz von Bücker jedoch in der Regel nicht gegeben. Zudem wird in der FMM mit dem beschriebenen Loadbalancing-Verfahren auf die Wechselwirkungen zwischen den Teilchen eingegangen. Diese Arbeit soll jedoch in erster Linie auf eine optimale Aufteilung der Gebiete eingehen, ohne dabei die Wechselwirkungen zwischen den Teilchen direkt zu berechnen.

2.5.2 Mehrstufiges eindimensionales Verfahren

Das zweite Loadbalancing-Verfahren, welches in dieser Arbeit zusätzlich betrachtet wird, ist das Verfahren von Halver. Dieses Verfahren wurde im Rahmen seiner Masterarbeit [7] am JSC erarbeitet.
Zu Beginn des Verfahrens wird das dreidimensionale System in äquidistante kubische Boxen eingeteilt. Zum Ausgleich der Teilchenanzahlen auf den einzelnen Prozessoren werden in diesem Verfahren nicht die Gitterpunkte verschoben, sondern Grenzen benachbarter Gebiete.

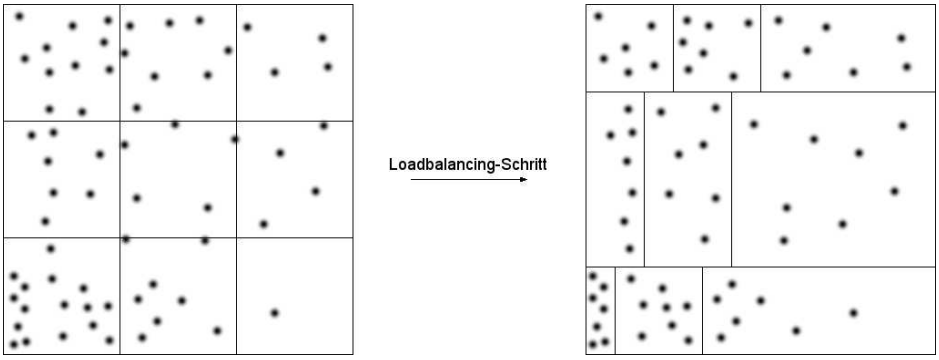


Abbildung 2.2: Aufteilung der Gebiete in der Arbeit von Halver, (Bildquelle:[7])

Zur Veranschaulichung zeigt Abbildung 2.2 ein zweidimensionales System, welches zu Beginn mit einem äquidistanten Gitter unterteilt ist. Der Schritt für das Loadbalancing besteht nun aus zwei wesentlichen Teilen. Zuerst werden die Gittergrenzen in y-Richtung angepasst und verschoben. Es entstehen drei Säulen unterschiedlicher Breite. Innerhalb dieser drei Säulen werden die Gittergrenzen unabhängig voneinander in x-Richtung verschoben.
Wenn man nun die 3. Dimension mit einbezieht, ergibt sich folgende Vorgehensweise. Die Anzahl der Grenzen, die in jeder Dimension verschoben werden dürfen, ist eindeutig durch die Anzahl der Gebiete festgelegt. Würde man das System beispielsweise in 27 Gebiete aufteilen, also drei Gebiete in jeder Richtung, so könnte man zwei Grenzen in jeder Richtung verschieben. Zuerst würde man in z-Richtung alle Ebenen verschieben, dann in y-Richtung und im letzten Schritt in der x-Richtung. Dabei wird die Verschiebung unabhängig von den anderen Dimensionen vorgenommen. Zuerst werden „Ebenen“ in z-Richtung angepasst, dann „Säulen“ in y-Richtung und schließlich Einzelgebiete in x-Richtung. Diese Verschiebung der Grenzen ist so zu wählen, dass die Arbeitsdifferenzen für die betrachteten Dimensionen ausgeglichen werden. Somit wird das eigentlich dreidimensionale Problem auf drei eindimensionale Probleme aufgeteilt, da die Verschiebung in jede der drei Richtungen unabhängig voneinander betrachtet wird. Die Verschiebung der Grenzen kann hier solange durchgeführt werden, bis alle Prozessoren in etwa die gleiche Arbeitslast haben. Durch die Tatsache, dass Grenzen verschoben werden und diese

Grenzen zusätzlich parallel zu den Seitenflächen sind, bleiben alle Gebiete quaderförmig. Dies hat den Vorteil, dass man sehr einfach und mit wenig Aufwand abfragen kann, ob ein Teilchen innerhalb oder außerhalb eines Gebietes liegt.
Darüber hinaus ist die Handhabung des Verfahrens sehr praktikabel, da man drei eindimensionale Probleme löst statt einem dreidimensionalen Problem. Ein Problem ist jedoch die Tatsache, dass Nachbarschaften nicht zwangsläufig erhalten bleiben. D.h. ein Prozessor bzw. ein Gebiet kann in unterschiedlichen Zeitschritten unterschiedliche Nachbarn haben. Diese Eigenschaft lässt sich mithilfe der Abbildung 2.2 nachvollziehen. Wenn die Grenzen der mittleren horizontalen Säule so verschoben werden, dass das obere linke Gebiet, das rechte Gebiet der mittleren Säule als Nachbargebiet erhält, so bleiben die ursprünglichen Nachbarschaftseigenschaften nicht erhalten. Diese Tatsache hat zur Folge, dass die Kommunikation aufwändiger und unübersichtlicher wird. Diese Problematik tritt in dem in dieser Arbeit beschriebenen Loadbalancing-Verfahren jedoch nicht auf, da jeder Prozessor in jedem Zeitschritt die gleichen Nachbarprozessoren und Nachbargebiete hat.

Kapitel 3

Verfahren zur Arbeitsverteilung und Domain Decomposition

In diesem Abschnitt wird die Vorgehensweise zur Gebietszerlegung für den zweidimensionalen und den dreidimensionalen Fall beschrieben. Dabei wird auf bestimmte Einschränkungen eingegangen, die berücksichtigt werden müssen, und die physikalischen und mathematischen Hintergründe näher betrachtet. Diese Überlegungen bieten die Grundlage für die Parallelisierung und die Gebietsaufteilung auf verschiedene Prozessoren. Das Ziel der Zerlegung besteht darin, eine möglichst optimale Aufteilung der Arbeit auf die Prozessoren zu erreichen. Dieses ist erreicht, falls die Unterschiede der zu verrichtenden Arbeiten, auf allen Prozessoren minimiert wurden. Ist diese Minimierung erreicht, so ist ein guter Ausgleich der Arbeit zwischen den Gebieten und Prozessoren gegeben und es kann eine gute parallele Auslastung erreicht werden.

3.1 Verfahren für zwei Dimensionen

Zu Beginn des Verfahrens ist das Gebiet in ein zweidimensionales äquidistantes Gitter aufgeteilt. In diesem Gitter werden die Teilchen, ihrer Ortskoordinate entsprechend, den Gebieten zugeordnet. Diese Gebiete werden im

Folgenden jedoch verändert, indem an den Eckpunkten der Gebiete mit einer bestimmten Kraft gezogen wird. Diese Kraft wird im Wesentlichen durch die Arbeit der angrenzenden Gebiete bestimmt. Die Arbeit der Gebiete kann man durch die verwendete Zeit pro Teilchen oder die Anzahl der Teilchen pro Gebiet definieren. Zur Bestimmung dieser Kräfte, auf Grundlage der Anzahl der Teilchen in jedem Gebiet, werden die folgenden Punkte abgearbeitet:

- Bestimmung der Massenschwerpunkte aller Gebiete:

$$\vec{r}^{(cm)} = \frac{1}{n} \cdot \sum_{i=1}^{n} \vec{r}_i$$

  wobei gilt:
  $\vec{r}_i$: Ortskoordinate des i-ten Teilchens auf dem betrachteten Gebiet.
  n: Anzahl der Teilchen auf dem Gebiet.
  $\vec{r}^{(cm)}$: Massenschwerpunkt des Gebietes.

- Berechnung der Kraft $\vec{f}_k$ mit der an dem Gitterpunkt k gezogen wird: Dazu werden die Massenschwerpunkte und die Anzahl der Teilchen aller vier angrenzenden Gebiete verwendet.

$$\vec{u}_i^0 = \frac{1}{\left|\left|\vec{r}_i^{(cm)} - \vec{r}_k\right|\right|} \cdot (\vec{r}_i^{(cm)} - \vec{r}_k)$$

  wobei gilt:
  $\vec{r}_i^{(cm)}$: Massenschwerpunkte der angrenzenden Gebiete.
  $\vec{r}_k$: Ortskoordinate des zu verschiebenden Gitterpunktes.
  $\vec{u}_i^0$: Richtungsvektor zwischen dem betrachteten Gitterpunkt k und dem Massenschwerpunkt des i-ten angrenzenden Gebiets in Richtung des Massenschwerpunktes, normiert auf die Länge eins.

- Berechnung des Kraftvektors $\vec{f}$, in dessen Richtung der Gitterpunkt verschoben werden soll:

$$\vec{f} = \sum_{i=1}^{4} (n_i - \overline{n}) \cdot \vec{u}_i^0 \tag{3.1}$$

Hier gilt:
n_i: Anzahl der Teilchen auf dem i-ten angrenzenden Gebiet, bezogen auf den zu überprüfenden Gitterpunkt.
$\overline{n}$: Mittelwert der Anzahl der Teilchen der vier angrenzenden Gebiete.

Wenn man die Arbeitsunterschiede in Richtung des größten Gradienten verschiebt, lässt sich die Verschiebung durch eine Kraft $\vec{f}$ ausdrücken:

$$\vec{x}^{(t+1)} = \vec{x}^{(t)} + \mu \cdot \vec{f}$$

Dabei ist die Variable μ ein Parameter, der am Anfang frei zu wählen ist, aber im Verlauf der Simulation adaptiv angepasst wird. Die Variable μ ist dabei ein Maß für die Schrittweite der Verschiebung, um eine optimale Arbeitsaufteilung zu erhalten. Welche Anpassungen durchgeführt werden können und welchen Einschränkungen die Schrittweite μ unterliegt wird im Laufe der Arbeit genauer untersucht.
Nachdem die Verschiebung der Gitterpunkte erfolgt ist, müssen die Teilchen neu in die Gebiete eingeteilt werden.
Da es sich nun nicht mehr um äquidistante Gitter handelt, muss die Abfrage auf die Zugehörigkeit der Teilchen anders formuliert werden. Zur Bearbeitung dieses Problems werden zwei Ansätze verfolgt:

1. Der erste Ansatz verwendet eine Rücktransformation der „Reference Shape Transformation“ [15] und wird im Kapitel 3.1.1 genauer beschrieben.

2. Die zweite implementierte Möglichkeit sieht vor, dass die Teilchen mittels Vektorrechnung auf ihre Gebietszugehörigkeit hin überprüft werden. Die genaue Vorgehensweise ist Kapitel 3.1.2 zu entnehmen.

3.1.1 Rücktransformation der „Reference Shape Transformation“

Die „Reference Shape Transformation“ ist eine Transformation, die von einem orthogonalen Referenz-System auf ein beliebig verzerrtes System abbildet. Hierbei werden im zweidimensionalen Fall vier Punkte aus dem quadratischen Gitter eindeutig auf vier Punkte im verzerrten physikalischen Gebiet abgebildet. Diese Transformation kann mit den folgenden Informationen und den im Folgenden beschriebenen Schritten erreicht werden.

Kennt man die vier Eckpunkte des verzerrten Systems

$$\xi^{(k)} = (\xi_1^{(k)}, \xi_2^{(k)})^T, \quad k \in \{1, 2, 3, 4\}$$

und die vier Punkte des Referenz-Systems,

$$x^{(k)} = \left(x_1^{(k)}, x_2^{(k)}\right)^T, k \in \{1, 2, 3, 4\}$$

so ist die einfachste Transformation eine Bilinear-Transformation, welche für $i \in 1, 2$ folgende Struktur hat:

$$\begin{pmatrix} \xi_1 \\ \xi_2 \end{pmatrix} = \begin{pmatrix} a_{01} + a_{11}x_1 + a_{21}x_2 + a_{31}x_1x_2 \\ a_{02} + a_{12}x_1 + a_{22}x_2 + a_{32}x_1x_2 \end{pmatrix} \in \mathbb{R}^{2\times 1} \tag{3.2}$$

Mithilfe der Koeffizienten a_{01}, a_{11}, a_{21}, a_{31}, a_{02}, a_{12}, a_{22}, a_{01} und a_{32} lässt sich diese Transformation in der folgenden kompakten Schreibweise darstellen:

$$\xi_i \quad = \sum_{j=0}^{3} a_{ji}\eta_j \tag{3.3}$$

Dazu werden die Koeffizienten in der Transformationsmatrix A zusammen-

gefasst:

$$A = \begin{pmatrix} a_{01} & a_{02} \\ a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{pmatrix} \in \mathbb{R}^{4\times 2} \tag{3.4}$$

Desweiteren gilt für den Vektor η:

$$\eta^T = (1, x_1, x_2, x_1 x_2)$$

Jeder Punkt $\xi^{(i)} = (\xi_1^{(i)}, \xi_2^{(i)})^T$ im verzerrten physikalischen System kann nun mithilfe der $x^{(i)} = \left(x_1^{(i)}, x_2^{(i)}\right)^T$ eindeutig dargestellt werden, wie in Gl. (3.2) angegeben.

Bis jetzt ist jedoch nur gesagt, dass es eine solche Bilinear-Transformation gibt, die aber aufgrund des gemischten Terms $x_1 x_2$ keine lineare Transformation ist. Im Folgenden wird gezeigt, wie die Koeffizienten a_{01}, a_{02}, a_{11}, a_{12}, a_{21}, a_{22}, a_{31} und a_{32} bestimmt werden können. Um dieses Ziel zu erreichen, werden im ersten Schritt die bekannten Koordinaten der Punkte des physikalischen Systems zu der Matrix Ξ zusammengefasst. Die Matrix W_η enthält dann die x-Koordinaten und die y-Koordinaten der vier Eckpunkte des physikalischen Systems:

$$\Xi = \begin{pmatrix} \xi_1^1 & \xi_2^1 \\ \xi_1^2 & \xi_2^2 \\ \xi_1^3 & \xi_2^3 \\ \xi_1^4 & \xi_2^4 \end{pmatrix} \in \mathbb{R}^{4\times 2}$$

Die Informationen der Eckpunkte des quadratischen Systems werden in der Matrix W_η so zusammengefasst, dass sie die vier Eckpunkte in der gleichen

Form wie η^T zeilenweise enthält:

$$W_\eta = \begin{pmatrix} 1 & x_1^{(1)} & x_2^{(1)} & x_1^{(1)}x_2^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & x_1^{(2)}x_2^{(2)} \\ 1 & x_1^{(3)} & x_2^{(3)} & x_1^{(3)}x_2^{(3)} \\ 1 & x_1^{(4)} & x_2^{(4)} & x_1^{(4)}x_2^{(4)} \end{pmatrix} \in \mathbb{R}^{4\times 4}$$

Mit der in Gl. (3.4) definierten Transformationsmatrix A lässt sich das folgende Gleichungssystem definieren:

$$\Xi = W_\eta A \tag{3.5}$$

Da man, weiß, welche Referenzpunkte auf welche physikalischen Punkte abgebildet werden, kann man durch einfaches Einsetzen die Matrizen Ξ und W_η bestimmen. Mit diesem Wissen lassen sich die Koeffizienten a_{ij} ohne weiteres über

$$A = W_\eta^{-1}\Xi$$

errechnen. Die Matrix W_η^{-1} kann bestimmt werden, da gefordert wurde, dass $\left(x_1^{(i)}, x_2^{(i)}\right)$ für $i \in \{1,2,3,4\}$ die vier Eckpunkte eines Quadrates sein müssen und die Matrix W_η^{-1} somit immer regulär ist. Die so bestimmten Koeffizienten der Matrix A können nun für alle Punkte, nicht mehr nur für die vier Eckpunkte, in der Gl. (3.3) zur Transformation verwendet werden.
Bis hier beschreibt das Verfahren eine Transformation von einem orthogonalen zweidimensionalen System in ein verzerrtes physikalisches Gebiet.
In dieser Masterarbeit sollen Teilchen auf ihre Gebietszugehörigkeit hin überprüft werden. Diese Gebiete sind dabei in verzerrter Form und nicht in orthogonaler Gestalt gegeben. Könnte man nun von einem verzerrten System in ein quadratisches System transformieren, so ist es sehr einfach zu entscheiden, ob ein Teilchen innerhalb oder außerhalb eines Gebietes liegt. Hier müssen dann nur die Koordinaten des Teilchens mit der oberen/unteren und der linken/rechten Grenze verglichen werden. Um dieses Ziel zu erreichen

muss nun jedes Teilchen vom physikalischen System in ein äquidistantes Gitter transformiert werden. Diese Transformation kann als Rücktransformation der in Gl. (3.2) beschriebenen Transformation gesehen werden, da nun die physikalischen Koordinaten gegeben sind. Die Koeffizienten der Matrix A sind ebenfalls gegeben, da die vier Eckpunkte beider Systeme bekannt sind. Demnach kann das Gleichungssystem aus Gl. (3.2) nach dem Vektor $(x_1, x_2)^T$ aufgelöst werden. Diese Lösung kann analytisch bestimmt werden und kann, falls die betrachteten Gebiete konvex sind, eindeutig dazu verwendet werden zu entscheiden, ob ein Teilchen innerhalb oder außerhalb eines Gebietes liegt.

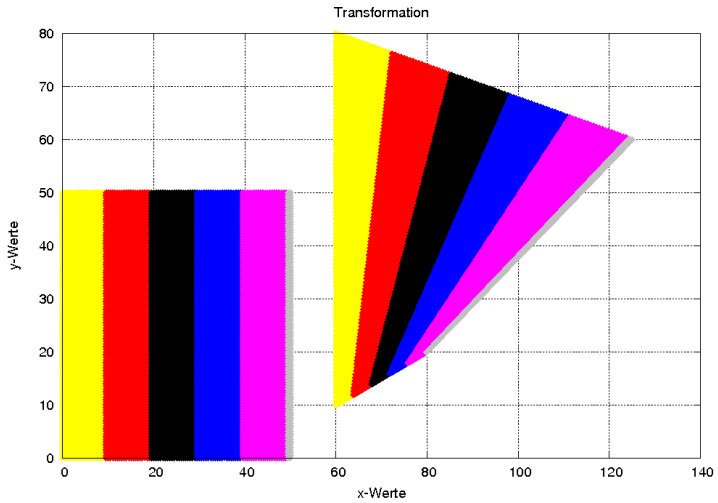


Abbildung 3.1: Transformation

Nach der Transformation auf das Referenz-System kann nun leichter mittels der Grenzen geprüft werden, ob das Teilchen innerhalb oder außerhalb des zu betrachtenden Gebietes liegt.
Die Abbildung 3.1 zeigt ein Beispiel, in dem von einem quadratischen Gebiet mit den Eckpunkten

$$(0|0)\,,\ (0|50)\,,\ (50|0)\,,\ (50|50)$$

in das verzerrte Gebiet mit den Eckpunkten

$$(60|10)\,,\ (60|80)\,,\ (80|20)\,,\ (125|60)$$

abgebildet wird. Bei der analytischen Lösung der Gl. (3.2) nach dem Vektor $(x_1, x_2)^T$ kommt es, wegen es gemischten Summanden $x_1 \cdot x_2$, zu zwei Lösungen, d.h. die Rücktransformation ist nicht eindeutig. In Abbildung 3.1 wurde von einem quadratischen Gebiet in ein verzerrtes Gebiet eindeutig abgebildet. In Abbildung 3.2 ist das verzerrte Gebiet grau dargestellt. Eine

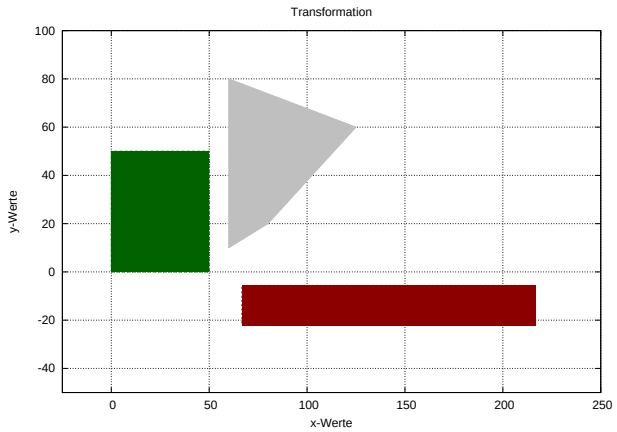


Abbildung 3.2: Rücktransformation von einem verzerrten Gebiet in zwei rechteckige Gebiete.

Lösung der Rücktransformation bildet von dem verzerrten Gebiet auf das rechteckige Gebiet ab, welches unterhalb des verzerrten Gebietes liegt. Die zweite Lösung bildet auf das quadratische Gebiet ab. Bei diesem handelt es sich um das ursprüngliche Gebiet, welches in Abbildung 3.1 zur Transformation genutzt wurde. Falls das verzerrte Gebiet konvex ist, so ist sichergestellt, dass die Lösung der Gl. (3.7) immer in das ursprüngliche quadratische Gebiet zurück abbildet. Ohne Beweis wird hier angegeben, dass Gl. (3.7) in das ursprüngliche Gebiet abbildet. In allen numerischen Testfällen, die in dieser Arbeit untersucht wurden, wurde dieses Verhalten bestätigt. Diese Lösung kann nun genutzt werden, um die Gebietszugehörigkeit leichter zu bestim-

men.

$$
\begin{aligned}
x_2 = \frac{1}{2} \cdot \Big(& a_{32} \cdot \xi_1 - a_{31} \cdot \xi_2 + (a_{02}a_{31} + a_{22}a_{11} - a_{01}a_{32} - a_{21}a_{12}) \\
& + \Big((2a_{32}^2 a_{01} + 2a_{02}a_{31}a_{32} + 2a_{11}a_{22}a_{32} + 2a_{32}a_{21}a_{21} + 4a_{22}a_{31}a_{12}) \cdot \xi_1 \\
& + (-2a_{31}^2 a_{02} + 2a_{31}a_{22}a_{11} + 2a_{31}a_{01}a_{32} + 2a_{31}a_{21}a_{12} - 4a_{21}a_{32}a_{11}) \cdot \xi_2 \\
& + a_{32}^2 \cdot \xi_1^2 + a_{31}^2 \cdot \xi_2^2 - 2a_{31}a_{32} \cdot \xi_1 \cdot \xi_2 \\
& + (a_{02}^2 a_{31}^2 + a_{22}^2 a_{11}^2 + a_{01}^2 a_{32}^2 + a_{21}^2 a_{32}^2 + a_{21}^2 a_{12}^2 \\
& - 2a_{02}a_{31}a_{22}a_{11} - 2a_{02}a_{31}a_{01}a_{32} - 2a_{02}a_{31}a_{21}a_{12} \\
& - 2a_{22}a_{11}a_{01}a_{32} - 2a_{11}a_{22}a_{21}a_{12} - 2a_{01}a_{32}a_{21}a_{12} \\
& + 4a_{22}a_{31}a_{01}a_{12} + 4a_{21}a_{32}a_{02}a_{11}) \Big)^{\frac{1}{2}} \frac{1}{a_{21}a_{32} - a_{22}a_{31}} \Big) \\
\wedge\ x_1 = & \frac{\xi_1 - a_{01} - a_{21}x_2}{a_{11} + a_{31}x_2}
\end{aligned}
\tag{3.6}
$$

$$
\begin{aligned}
x_2 = -\frac{1}{2} \cdot \Big(& a_{31} \cdot \xi_2 - a_{32} \cdot \xi_1 + (a_{01}a_{32} + a_{21}a_{12} - a_{02}a_{31} - a_{22}a_{11}) \\
& + \Big((2a_{32}^2 a_{01} + 2a_{02}a_{31}a_{32} + 2a_{11}a_{22}a_{32} + 2a_{32}a_{21}a_{21} + 4a_{22}a_{31}a_{12}) \cdot \xi_1 \\
& + (-2a_{31}^2 a_{02} + 2a_{31}a_{22}a_{11} + 2a_{31}a_{01}a_{32} + 2a_{31}a_{21}a_{12} - 4a_{21}a_{32}a_{11}) \cdot \xi_2 \\
& + a_{32}^2 \cdot \xi_1^2 + a_{31}^2 \cdot \xi_2^2 - 2a_{31}a_{32} \cdot \xi_1 \cdot \xi_2 \\
& + (a_{02}^2 a_{31}^2 + a_{22}^2 a_{11}^2 + a_{01}^2 a_{32}^2 + a_{21}^2 a_{32}^2 + a_{21}^2 a_{12}^2 \\
& - 2a_{02}a_{31}a_{22}a_{11} - 2a_{02}a_{31}a_{01}a_{32} - 2a_{02}a_{31}a_{21}a_{12} \\
& - 2a_{22}a_{11}a_{01}a_{32} - 2a_{11}a_{22}a_{21}a_{12} - 2a_{01}a_{32}a_{21}a_{12} \\
& + 4a_{22}a_{31}a_{01}a_{12} + 4a_{21}a_{32}a_{02}a_{11}) \Big)^{\frac{1}{2}} \frac{1}{a_{21}a_{32} - a_{22}a_{31}} \Big) \\
\wedge\ x_1 = & \frac{\xi_1 - a_{01} - a_{21}x_2}{a_{11} + a_{31}x_2}
\end{aligned}
\tag{3.7}
$$

Das System dieser Rücktransformation der „Reference Shape Transformation“ ist wie beschrieben analytisch nach $(x_1, x_2)^T$ auflösbar.
Im dreidimensionalen Fall kann man kann man ebenfalls eine analoge Transformation definieren. Wenn die Koeffizienten der Matrix A und die Koordinaten des Punktes $x = (x_1, x_2, x_3)^T$ aus dem Referenz-System bekannt sind,

so geschieht die Transformation analog zu dem zweidimensionalen Fall:

$$\begin{pmatrix}\xi_1\\ \xi_2\\ \xi_3\end{pmatrix} = \begin{pmatrix}a_{01}+a_{11}x_1+a_{21}x_2+a_{31}x_3+a_{41}x_1x_2+a_{51}x_1x_3+a_{61}x_2x_3+a_{71}x_1x_2x_3\\ a_{02}+a_{12}x_1+a_{22}x_2+a_{32}x_3+a_{42}x_1x_2+a_{52}x_1x_3+a_{62}x_2x_3+a_{72}x_1x_2x_3\\ a_{03}+a_{13}x_1+a_{23}x_2+a_{33}x_3+a_{43}x_1x_2+a_{53}x_1x_3+a_{63}x_2x_3+a_{73}x_1x_2x_3\end{pmatrix}$$

Zur Bestimmung der Koeffizienten der Matrix A benötigt man nun acht Eckpunkte aus dem Referenz-System und acht Punkte aus dem physikalischen System. Für das System

$$\Xi = W_\eta A$$

welches unter (3.5) bereits für den zweidimensionalen Fall definiert wurde, ergeben sich für die Matrizen die folgenden Dimensionen:

$$\Xi \in \mathbb{R}^{8\times 3}, W_\eta \in \mathbb{R}^{8\times 8}, A \in \mathbb{R}^{8\times 8}$$

Die Koeffizienten der Matrix A können analytisch bestimmt werden. Problematisch wird es jedoch bei der Rücktransformation, wenn man versucht das Gleichungssystem für drei Dimensionen nach $x = (x_1, x_2, x_3)^T$ aufzulösen. Die Variable x lässt sich aufgrund des kubischen Terms $x_1x_2x_3$ und der Tatsache dass es sich um gekoppelte nicht-lineare Gleichungen handelt, nicht ohne Weiteres bestimmen und sich auch nicht effizient implementieren. Für den dreidimensionalen Fall wird diese Rücktransformation in dieser Arbeit jedoch nicht weiter berücksichtigt. Im dreidimensionalen Fall wird die Gebietszugehörigkeit ausschließlich über Vektorrechnung bestimmt.

3.1.2 Vektorbasierter Test auf Gebietszugehörigkeit

Eine weitere Möglichkeit zu überprüfen, ob ein Teilchen innerhalb oder außerhalb eines Gebietes liegt, bietet die Vektorrechnung. Der hier verwendete Ansatz für den zweidimensionalen Fall kann im Wesentlichen ebenfalls für den dreidimensionalen Fall übernommen werden.
Bekannt ist, dass jedes Gebiet von seinen vier Eckpunkten eindeutig bestimmt ist. Somit können vier Vektoren ($\vec{u}_1,\ \vec{u}_2,\ \vec{u}_3,\ \vec{u}_4$), die das Gebiet begrenzen über die Differenz der Eckpunkte ($\vec{e}_1,\ \vec{e}_2,\ \vec{e}_3,\ \vec{e}_4$) berechnet

werden.

$$\vec{u}_1 = \vec{e}_2 - \vec{e}_1$$

$$\vec{u}_2 = \vec{e}_3 - \vec{e}_2$$

$$\vec{u}_3 = \vec{e}_4 - \vec{e}_3$$

$$\vec{u}_4 = \vec{e}_1 - \vec{e}_4$$

Im nächsten Schritt bestimmt man die Normalenvektoren $\vec{n}_1$, $\vec{n}_2$, $\vec{n}_3$ und $\vec{n}_4$ in dem man die Komponenten der Richtungsvektoren vertauscht und den x-Wert mit einem Vorzeichenwechsel versieht. Diese vier Vektoren stehen senkrecht auf den begrenzenden Vektoren ($\vec{u}_1$, $\vec{u}_2$, $\vec{u}_3$, $\vec{u}_4$) und werden so definiert, dass sie nach außen zeigen. Nun wird für jede der vier Seiten

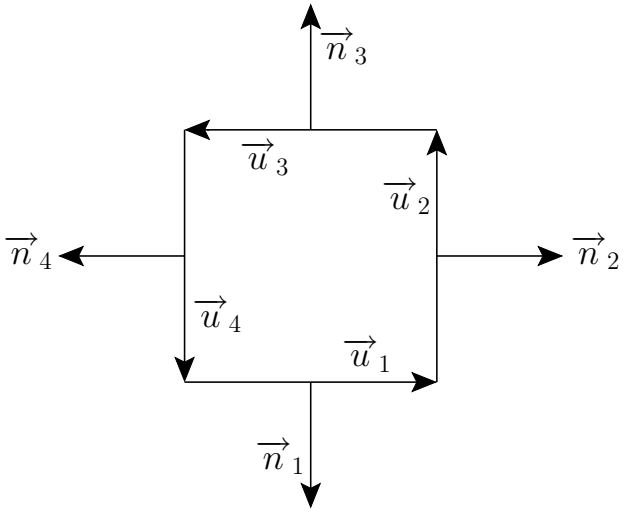


Abbildung 3.3: Test auf Gebietszugehörigkeit

der Verbindungsvektor zwischen dem zu testenden Teilchen $\vec{r}_t$ und einem Punkt auf jeder Geraden $\vec{q}_{1..4}$ ermittelt. Das Skalarprodukt zwischen diesem Differenzvektor und dem zu dieser Seite gehörigen Normalenvektor, welches in Gl. (3.8) steht, gibt Informationen darüber, ob sich das Teilchen im Gebiet befindet oder nicht.

$$\langle (\vec{r}_t - \vec{q}_i), \vec{n}_i \rangle \tag{3.8}$$

Setzt man die Definition des Skalarproduktes ein so erhält man:

$$\langle (\vec{r}_t - \vec{q}_i), \vec{n}_i \rangle = ||\vec{r}_t - \vec{q}_i|| \cdot ||\vec{n}_i|| \cdot \cos(\alpha)$$

Die Längen $||\vec{r}_t - \vec{q}_i||$ und $||\vec{n}_i||$ sind positive reelle Zahlen. Der Winkel α bezeichnet den Winkel zwischen dem Differenzvektor, welcher in Richtung des zu testenden Teilchens t zeigt, und dem Normalenvektor. Der Winkel α bestimmt nun das Vorzeichen des Skalarproduktes:

$$0 < \alpha < \frac{\pi}{2} \Rightarrow \qquad \cos(\alpha) > 0 \Rightarrow \qquad \langle (\vec{r}_t - \vec{q}_i), \vec{n}_i \rangle > 0 \tag{3.9}$$

$$\frac{\pi}{2} < \alpha < \pi \Rightarrow \qquad \cos(\alpha) < 0 \Rightarrow \qquad \langle (\vec{r}_t - \vec{q}_i), \vec{n}_i \rangle < 0 \tag{3.10}$$

Für den Fall (3.9) ist der eingeschlossene Winkel zwischen dem Verbindungsvektor und dem Normalenvektor kleiner als $\frac{\pi}{2}$. Dies ist genau dann gegeben, wenn das Teilchen auf der Seite liegt zu welcher der Normalenvektor zeigt. Da der Normalenvektor aus dem betrachteten Gebiet heraus zeigt, liegt das Teilchen für diese Seite außerhalb des Gebietes. Liegt der Winkel α wie in der Gl. (3.10) zwischen $\frac{\pi}{2}$ und π, so liegt das Teilchen auf der Seite der Geraden, die sich innerhalb des Gebietes befindet. Falls nun die Skalarprodukte für alle vier Seiten negativ sind, so liegt das betrachtete Teilchen innerhalb des Gebietes. Lag das Teilchen im vorrangegangenen Schritt noch innerhalb des Gebiets und hat das Gebiet nun verlassen, so lässt sich über die Vorzeichen der einzelnen Skalarprodukte zusätzlich sagen über welche Seite das Teilchen in ein anderes Gebiet gewandert ist. Die Behandlung der Teilchen, die ein Gebiet verlassen haben, wird in Kapitel 3.2.1 genauer erläutert.
Die bis hier beschriebene Idee der Gebietszugehörigkeit mittels Vektorrechnung kann im Wesentlichen auf den dreidimensionalen Fall übertragen werden. Dort werden dann nicht vier Seitenkanten, sondern sechs bzw. zwölf Seitenflächen betrachtet. Die genaue Beschreibung dazu ist in Kapitel 3.2.3 zu finden.

3.2 Verallgemeinertes Verfahren

Dieses Kapitel beschäftigt sich mit der Frage, wie mit Teilchen verfahren wird, die ein Gebiet verlassen haben. Dieses Vorgehen ist für den zwei- und dreidimensionalen Fall sehr ähnlich. Außerdem sind die Behandlungen der Gitterpunkte am Rand des Systems ebenfalls sehr ähnlich für zwei und drei Dimensionen. Im zweiten Teil dieses Kapitels werden die dafür verwendeten Projektionen beschrieben und näher auf die Randbedingungen eingegangen.

3.2.1 Transfer an benachbarte Gebiete

Bleibt ein Teilchen nach seiner Verschiebung oder nach der Verschiebung der Gitterpunkte im gleichen Gebiet, so wird dieses Teilchen an dieser Stelle nicht weiter berücksichtigt. Handelt es sich jedoch um ein Teilchen, welches das Gebiet verlassen hat, so wird in einem ersten Schritt geprüft, ob das Teilchen das Prozessor-Gebiet nach rechts oder links verlassen hat. Ist dies der Fall, so wird es dem rechten bzw. linken Gebiet zugeordnet. Die Nachbargebiete werden zu Beginn zugeordnet und bleiben im weiteren Verlauf der Simulation erhalten. In Schritt zwei wird die Verschiebung nach oben und unten betrachtet. Ist ein Teilchen nach oben oder unten herausgelaufen, so wird es an den jeweiligen Prozessor weitergegeben. Diese Weitergabe der Teilchen ist ohne weiteres möglich da die Nachbarschaftsbeziehungen zwischen den Gebieten bzw. Prozessoren immer erhalten bleiben. Wenn man davon ausgeht, dass ein Teilchen nicht so weit verschoben wird, dass es über zwei Gebiete läuft, können alle Nachbargebiete, wie in Abbildung 3.4 zu erkennen, erreicht werden. In drei Dimensionen werden die Teilchen ebenfalls zuerst nach rechts und links weiter gegeben, dann nach oben und unten. Zusätzlich werden die Teilchen dann aber noch in die dritte Dimension übergeben, falls sie das Gebiet in dieser Dimension verlassen haben. In diesem Fall würde man nicht nur die Nachbargebiete, welche in der Ebene liegen erreichen, sondern alle Nachbargebiete im Raum. Für die Abbildung 3.4 würde dies heißen, dass man einen Würfel erhalten würde.

Jedes Teilchen besitzt Informationen über seinen Ort und seine Geschwin-

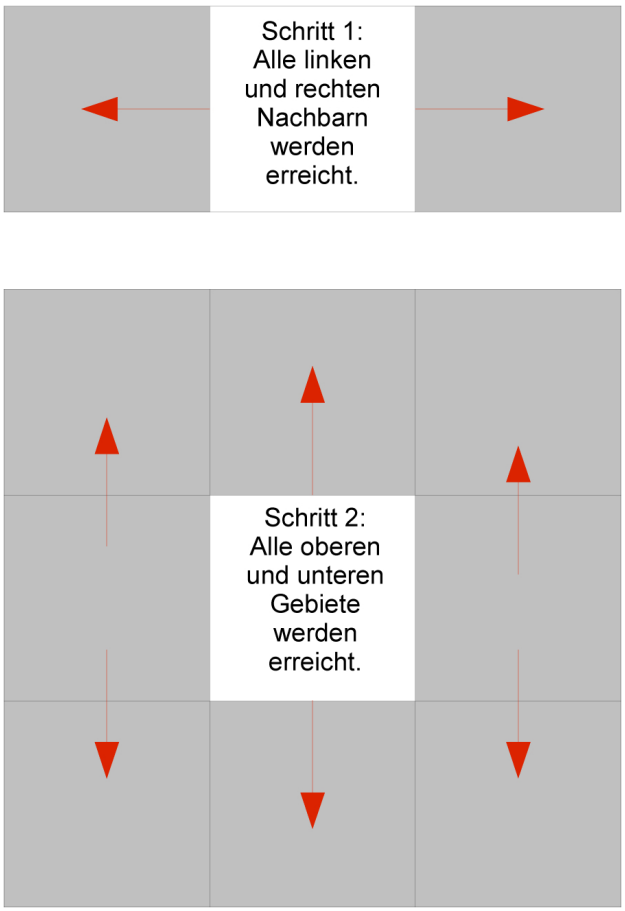


Abbildung 3.4: Weitergabe der Teilchen

digkeit. Auf jedem Prozessor sind die Teilcheninformationen in sequentiellen Listen bzw. Arrays gespeichert. Um die Teilcheninformationen optimal verarbeiten zu können ist es wichtig, dass diese Listen zusammenhängend sind, d.h. keine „Löcher“ beinhalten. Wenn ein Teilchen das Gebiet des Prozessors verlassen hat und an einen der Nachbarprozessoren verschickt wurde, so entsteht an dieser Stelle des Arrays eine Lücke. Abbildung 3.5 zeigt, wie diese Lücken geschlossen werden. Zur Vereinfachung wird davon ausgegangen, dass fünf Teilchen auf dem betrachteten Prozessor liegen. D.h. die Stellen 1 bis 5 sind zu Beginn belegt. Durch die Verschiebung der Gittergrenzen bekommt das Gebiet drei neue Teilchen hinzu. Außerdem werden die Teilchen an den Stellen 1 und 3 abgegeben. Da hier nur der Algorithmus zum Einsortieren der

Teilchen in die Listen gezeigt werden soll ist es unerheblich wohin die Teilchen abgegeben werden und woher man sie erhält. Zusätzlich zu dem Feld, in dem die Teilchen mit ihren Informationen abgespeichert sind, wird ein Feld angelegt, welches besagt, ob ein Teilchen abgegeben wurde oder nicht. Sollte ein Teilchen abgegeben worden sein und keine Relevanz mehr für den aktuellen Prozessor haben, so wird das Flag in dem Zusatzfeld auf den Wert 1 gesetzt. Die Summe über alle Felder des Zusatzfeldes ergibt dann direkt die Anzahl der abgegebenen Teilchen, in diesem Beispiel $n = 2$. Im nächsten Schritt werden die drei neuen Teilchen an das Ende der Liste angefügt. Im letzten Schritt werden von hinten nach vorne die letzten n Teilchen der Reihe nach an den freien Stellen der Liste einsortiert. Diese freien Stellen sind jetzt mithilfe des Zusatzfeldes einfach zu bestimmen. Der Prozessor für das in der Abbildung 3.5 gezeigte Beispiel hat nun sechs Teilchen, die ohne Lücken hintereinander im Array sich befinden. Der hier beschriebene Algorithmus funktioniert auch dann, wenn mehr Teilchen vom Prozessor abgegeben werden, als hinzu kommen. Dann werden Teilchen, die bereits vorher auf auf dem Prozessor lagen, von hinten in die frei gewordenen Plätze des Arrays einsortiert, bis die freien Plätze belegt sind.

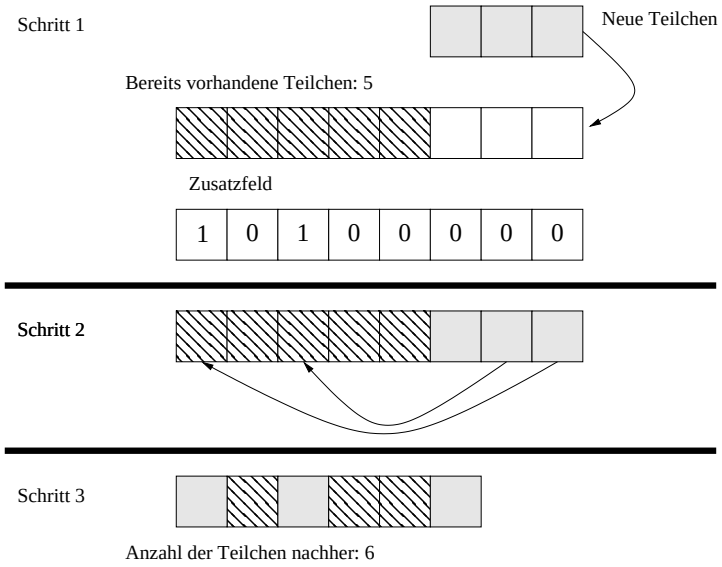
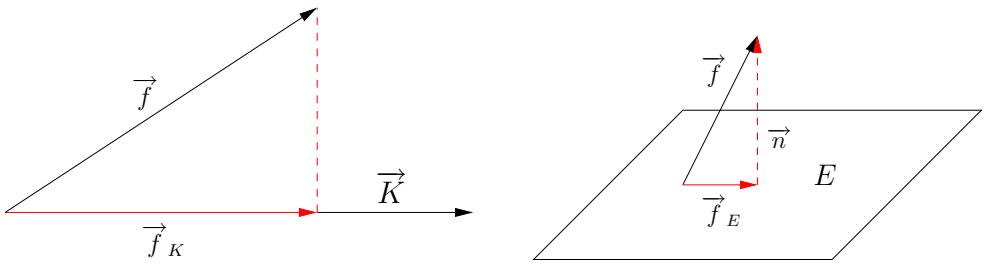


Abbildung 3.5: Beispiel zum Einsortieren der Teilchen auf einem Prozessor

3.2.2 Die orthogonale Projektion und periodische Randbedingungen

Bis jetzt wurde nur beschrieben, wie die Verschiebung von inneren Gitterpunkten durchgeführt wird. Trotz der Tatsache, dass insbesondere bei einer großen Anzahl an Gebieten und Prozessoren der Anteil der inneren Gitterpunkte sehr hoch ist, muss man sich damit beschäftigen, wie die Gitterpunkte am Rand des Systems verschoben werden. Wenn ein Punkt auf einer Seitenkante oder auf einer Seitenfläche des Gebiets liegt, so soll er auch nach der Verschiebung immer auf derselben Kante bzw. Fläche liegen. Zur Berechnung der Kraft, die auf die „Randgitterpunkte“ wirkt, werden bestimmte Randbedingungen angenommen. Diese Randbedingungen und die orthogonale Projektion der Kräfte, die dafür sorgt, dass keine „Randgitterpunkte“ aus ihrer Kante bzw. Ebene herausgezogen werden, werden in diesem Kapitel beschrieben und erklärt. Die Ergebnisse dieses Kapitels gelten gleichermaßen für den zweidimensionalen und den dreidimensionalen Fall.
Jeder Gitterpunkt besitzt im zweidimensionalen Fall vier und im dreidimensionalen Fall acht angrenzende Gebiete. Für Randpunkte gilt dies erst mithilfe von periodischen Randbedingungen, die besagen, dass Teilchen, die rechts aus dem System herauslaufen auf der linken Seite wieder genauso ins System eintreten. So kann man nachvollziehen, dass dann auch „Randgitterpunkte“ an vier bzw. acht Gebiete angrenzen. Ein Gitterpunkt der beispielsweise auf der unteren Fläche des Systems liegt hat im dreidimensionalen Fall bis zu vier Nachbargebiete, die unmittelbar oberhalb von ihm liegen. Außerdem besitzt er vier weitere Nachbargebiete, die graphisch unterhalb des Gitterpunktes liegen. Für diese vier Gebiete werden die vier obersten Gebiete in dieser Säule imaginär unterhalb des zu berechnenden Gitterpunktes gesetzt. Diese periodische Randbedingung gilt für alle Dimensionen. Die resultierenden Kraftvektoren der „Randgitterpunkte“, deren Berechnung unter der Annahme der periodischen Randbedingungen analog zu den inneren Punkten abläuft, können von der Randseite aus dem System heraus oder in das System hinein zeigen. Eine Verschiebung des Punktes entlang dieses Vektors hätte jedoch zur Folge, dass das Gebiet des gesamten Systems verzerrt werden würde. Um



(a) Projektion auf einen Vektor (b) Projektion auf eine Ebene

Abbildung 3.6: Allgemeine Projektionen

eine solche Verzerrung zu verhindern, werden die Eckpunkte des Systems gar nicht bewegt und für alle anderen „Randgitterpunkte“ werden die Kraftvektoren auf die Seitenfläche bzw. Seitenkante projiziert. Im zweidimensionalen Fall reicht es aus, die orthogonale Projektion auf Seiten bzw. Kanten zu berücksichtigen. Im dreidimensionalen Fall werden die Kräfte der „Randgitterpunkte“ die auf einer Seitenkante des Systems liegen auch auf diese Kanten projiziert. Zusätzlich gibt es aber auch den Fall, dass resultierende Kraftvektoren auf Ebenen projiziert werden müssen. Dies gilt für „Randgitterpunkte“, die sich auf Seitenebenen, aber nicht auf Seitenkanten befinden. Abbildung 3.6 zeigt die allgemeine Projektion eines Vektors auf einen anderen Vektor und die Projektion eines Vektors auf eine Ebene, wie sie in dem Programmpaket dieser Arbeit verwendet werden. Nach den obigen Überlegungen zu der orthogonalen Projektion und den periodischen Randbedingungen lässt sich der mathematische Kern leicht über die analytische Geometrie definieren.

Die Kraft $\vec{f}$ ist die Kraft, die auf den Gitterpunkt einer Seitenkante wirkt, wenn man ausschließlich die Berechnung für die Kraft zur Verschiebung der Gitterpunkte aus der Gl. (3.1) berücksichtigt. Diese Kraft soll nun aber entlang der Seitenkante $\vec{K}$ wirken. Zu diesem Zweck wird $\vec{f}$ auf $\vec{K}$ projiziert(Abbildung 3.6(a)). Diese Projektion, in der Formel (3.11) mit $\vec{f}_K$ bezeichnet, gibt an in welche Richtung der Gitterpunkt verschoben wird, ohne dabei die Form des Systems zu verändern.

$$\vec{f}_K = \vec{K} \frac{\left\langle \vec{f}, \vec{K} \right\rangle}{\left|\left| \vec{K} \right|\right|^2} \tag{3.11}$$

Wie bereits erwähnt, muss die resultierende Kraft für Gitterpunkte, die auf einer Seitenfläche aber nicht auf einer Seitenkante liegen auf genau diese Seitenfläche projiziert werden. Die Abbildung 3.6(b) zeigt die Projektion der Kraft $\vec{f}$ auf die Ebene E. Diese Projektion geschieht mittels der folgenden Formel:

$$\vec{f}_E = \vec{f} - \frac{\left\langle \vec{f} - \vec{a}, \vec{n} \right\rangle}{||\vec{n}||^2} \vec{n} \tag{3.12}$$

In Gl. (3.12) beschreibt $\vec{f}_E$ den projizierten Vektor der Kraft, in dessen Richtung der Gitterpunkt verschoben wird und $\vec{a}$ den Ortsvektor zu einem beliebigen Punkt auf der Ebene. Darüber hinaus ist $\vec{n}$ ein Normalenvektor der Ebene E.

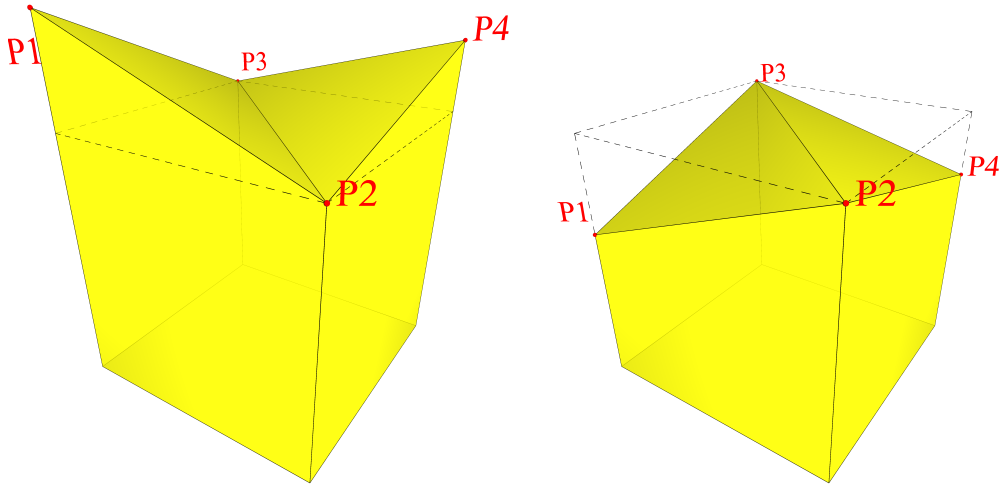
Unter Berücksichtigung dieser Projektionen und der Verschiebung der Randgitterpunkte in Richtung von $\vec{f}_K$ oder $\vec{f}_E$, ausgenommen sind die vier bzw. acht Eckpunkte des Systems ist sichergestellt, dass die äußere Form des Systems erhalten bleibt.

3.2.3 Verfahren für drei Dimensionen

Das Grundprinzip und die Vorgehensweise zur Bestimmung der Prozessorgebiete baut auf den Überlegungen des zweidimensionalen Falls auf und beinhaltet am Rand die in Kapitel 3.2.2 beschriebene Vorgehensweise. Da nun an jeden Gitterpunkt acht Gebiete angrenzen, statt vier Gebiete wie im zweidimensionalen Fall, wird an diesem mit acht verschiedenen Kräften gezogen, aus denen die resultierende Kraft berechnet wird. Die Berechnungen der Massenschwerpunkte, der Kräfte der inneren Punkte und der mittleren Anzahl

der Teilchen der angrenzenden Gebiete geschieht völlig analog zu den Berechnungen aus dem zweidimensionalen Fall.
Bei der Berechnung der Kräfte mit denen an den seitlichen Punkten gezogen wird treten jedoch Unterschiede auf. Ein Punkt der an der Seitenkante liegt darf auch nur auf dieser Seitenkante verschoben werden. Ein Punkt der auf einer der Seitenflächen aber nicht auf einer Seitenkante liegt, darf nur auf dieser Fläche verschoben werden. Somit muss die Kraft, die an einem Gitterpunkt, der sich auf einer seitlichen Kante befindet, auf eben diese Kante projiziert werden. Die Vorgehensweise ist Kapitel 3.2.2 zu entnehmen.
Für den Test, ob ein Teilchen außerhalb oder innerhalb eines Gebietes liegt wird zudem folgende Erweiterung betrachtet: Im zweidimensionalen Fall wurde das Gebiet zu einer Seite hin eindeutig durch die Gerade zwischen zwei Punkten begrenzt. Im dreidimensionalen Fall wird das betrachtete Gebiet zu einer Seite hin durch vier Punkte begrenzt. Da diese vier Punkte im Allgemeinen nicht in einer Ebene liegen, wird die Gebietsbegrenzung zu einer Seite durch das Aufspannen von zwei Ebenen bestimmt.
Die Vorgehensweise für den Test auf Gebietszugehörigkeit wird an dieser Stelle mithilfe der Abbildung 3.7 gezeigt. Die Gitterpunkte $\vec{p}_1$, $\vec{p}_2$, $\vec{p}_3$ und $\vec{p}_4$ begrenzen das gelbe Gebiet nach oben. Dabei spannen die Punkte $\vec{p}_1$, $\vec{p}_2$ und $\vec{p}_3$ die erste begrenzende Ebene und die Punkte $\vec{p}_2$, $\vec{p}_3$ und $\vec{p}_4$ die zweite begrenzende Ebene auf. An der oberen Begrenzung wird nun exemplarisch gezeigt, wie erkannt werden kann, ob ein Teilchen das Gebiet nach oben hin verlassen hat. Dazu werden zwei Fälle unterschieden.
Es kann sein, dass die obere Fläche bezogen auf das Gebiet konkav ist, siehe dazu 3.7 (a). In diesem Fall liegt ein Teilchen genau dann innerhalb des Gebietes, wenn es unterhalb einer der beiden Ebenen liegt, da es genau dann in das gelbe Gebiet fällt, vorrausgesetzt, dass das betrachtete Teilchen zu keiner anderen Seite herausgelaufen ist. Das betrachtete Teilchen liegt nur dann oberhalb des Gebietes, falls es oberhalb der beiden Ebenen liegt. Diese Vorgehensweise liefert ein eindeutiges Kriterium, wenn die Begrenzung zu dieser Seite hin konvex ist.
Ist das Gebiet nach oben hin beschränkt, wie in Abbildung 3.7 (b) zu sehen, so wird die Beschränkung in diese Richtung als konkav definiert. D.h.

Die Verbindungsgerade zwischen den Punkten $\vec{p}_2$ und $\vec{p}_3$, welche in beiden Ebenen liegt, liegt nun oberhalb der Punkte $\vec{p}_1$ und $\vec{p}_4$. Bei dieser Art der Beschränkung des Gebietes liegt ein Teilchen noch innerhalb des Gebietes, falls es unterhalb der Ebene liegt die von $\vec{p}_1$, $\vec{p}_2$ und $\vec{p}_3$ aufgespannt wird und unterhalb der Ebene, die durch die Punkte $\vec{p}_2$, $\vec{p}_3$ und $\vec{p}_4$. Auch hier wird vorrausgesetzt, dass das Teilchen das Gebiet zu keiner anderen Seite hin verlassen hat. Sobald sich das zu testende Teilchen oberhalb einer der beiden Ebenen befindet, liegt das Teilchen nicht mehr im „gelben“ Gebiet, sondern hat es nach oben verlassen.
Die Abfrage, ob ein Teilchen oberhalb oder unterhalb der betrachteten Ebenen liegt geschieht entsprechend der Herangehensweise im zweidimensionalen Fall. Dies geschieht über die Definition des Skalarproduktes und den Normalenvektoren, die jeweils senkrecht auf den einzelnen Ebenen stehen. Da man die Überlegungen zu Gl. (3.9) und Gl. (3.10) so, wie in Kapitel 3 beschrieben, im dreidimensionalen Fall übernehmen kann werden diese an dieser Stelle nicht weiter erläutert.
Die hier beschriebene Vorgehensweise wird ebenfalls für die anderen sechs Seiten vorgenommen. Ein Teilchen liegt nur dann in einem Gebiet, wenn für alle sechs Seiten das Teilchen als „innerhalb“ erkannt wurde. Ist dies für mindestens eine Seite nicht der Fall so kann das Teilchen nicht mehr im Gebiet liegen und es muss an einen Nachbarprozessor verschickt werden.
Zwei benachbarte Gebiete haben immer eine gemeinsame Grenzfläche, bzw. zwei gemeinsame begrenzende Dreiecke. Wichtig ist, dass diese Ebenen in beiden Gebieten gleich definiert sind, damit die Gebiete der unterschiedlichen Prozessoren disjunkt sind. Bezogen auf die Abbildung 3.7 bedeutet das, dass für den Fall (a) das oben angrenzende Gebiet nach unten hin konvex ist. Für den Fall (b) wäre das obenliegende Gebiet nach unten hin konkav, damit die Disjunktheit dieser beiden Gebiete gegeben ist.



(a) Konkave Begrenzung (b) Konvexe Begrenzung

Abbildung 3.7: Begrenzung des Gebietes nach oben

3.3 Stabilität des Verfahrens

Die einzige Einschränkung an die Gebiete der einzelnen Prozessoren besteht darin, dass alle diese Gebiete immer konvex bleiben sollen. Abbildung 3.8 zeigt ein beliebiges Gebiet eines Prozessors. Die Eckpunkte dieses Gebietes werden von verschiedenen Prozessoren unabhängig verschoben. Wenn man keine Bedingungen bzw. Einschränkungen für die Verschiebung der Gitterpunkte angibt könnte es passieren, dass die Gitterpunkte so verlaufen, wie es die Pfeile in Abbildung 3.8 andeuten. Dieses Verhalten der Gitterpunkte wäre fatal, da es dann zu Durchdringungen kommen würde und keine Gebiete mit zusammenhängenden Flächen mehr existieren würden. Dieser Fall kann nicht eintreten, falls man ausschließt, dass ein Gitterpunkt die Diagonale (hier gestrichelt eingezeichnet) überschreitet. Auf dieser Überlegung basiert das im Folgenden beschriebene Stabilitätskriterium. Kein Punkt darf eine Diagonale seiner angrenzenden Gebiete überschreiten. Dies hat zu Folge, dass alle Gebiete in einem zweidimensionalen System konvex sind.

Anschaulich heißt dies, dass die Verbindungsgeraden aller Punkte des Vierecks vollständig innerhalb des Gebietes liegen. Wie in der Abbildung 3.9 zu

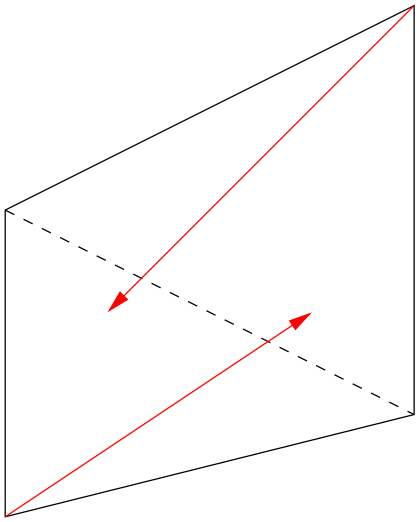


Abbildung 3.8: Nicht zulässiger Verlauf der Gitterpunkte

erkennen, ist es bei einem konkaven Viereck möglich, dass die Verbindungsgerade zweier innerer Punkte nicht vollständig im betrachteten Gebiet liegt.

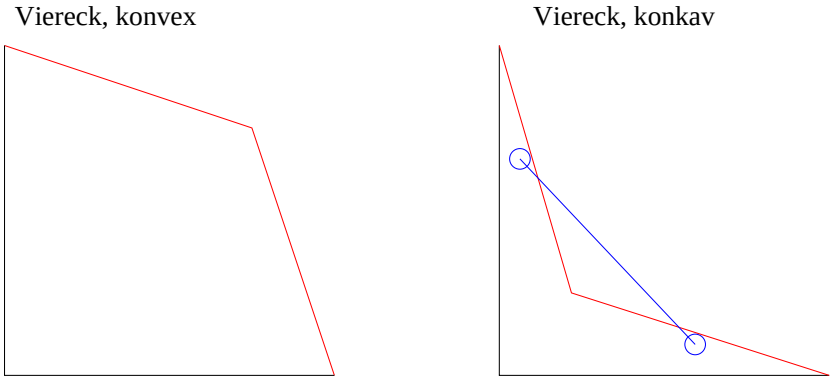
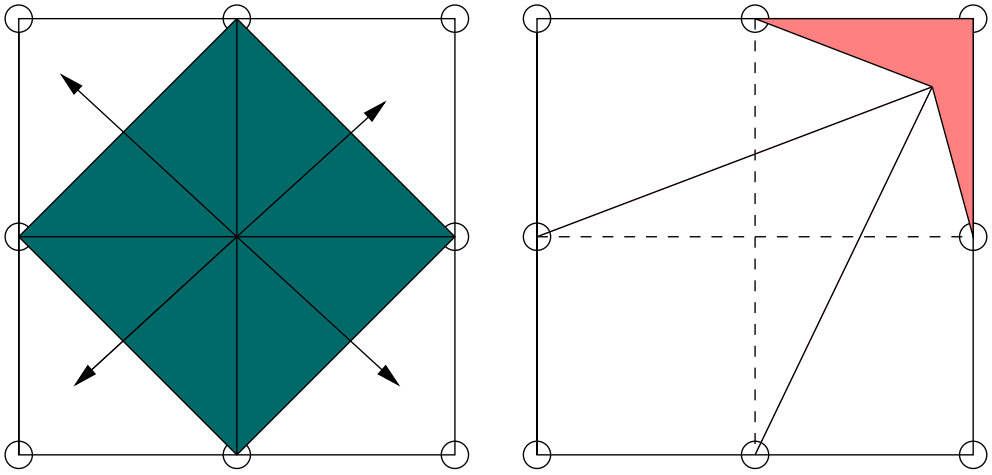


Abbildung 3.9: Konvexes und konkaves Viereck

Wenn man Konvexität aller Gebiete fordert wird zudem sichergestellt, dass es nicht zu einem entarteten Viereck kommen kann. D.h., es ist nicht möglich, dass einer der vier Gitterpunkte sich so verschiebt, dass er auf der Verbindungsgeraden der zwei benachbarten Punkte, also der Diagonalen des Vierecks, liegt. Könnte dieser Fall eintreten, so würde man faktisch kein Viereck sondern nur noch ein Dreieck untersuchen. Da aber alle Überlegungen auf Vierecken beruhen, wird dieser Fall ausgeschlossen.



(a) Die farbige Fläche darf nicht verlassen werden.

(b) Ein konkaves Gebiet, falls das Kriterium nicht eingehalten wird.

Abbildung 3.10: 1. Kriterium zur Konvexität

Um die Konvexität der Vierecke sicherzustellen, werden die im Folgenden beschriebenen Schritte berücksichtigt. Zusätzlich ist es möglich eine optionale Kraft zu definieren, die einen Gitterpunkt von den gegenüberliegenden Diagonalen in Abhängigkeit des Abstandes abstoßen lässt. Anschaulich bedeutet das Kriterium für einen Gitterpunkt, dass er nach seiner Verschiebung nicht die farbige Fläche in Abbildung 3.10(a) verlässt. Würde der mittlere Gitterpunkt diese farbige Fläche verlassen, so würde mindestens ein nicht konvexes Viereck entstehen. Ein Beispiel dazu zeigt die Abbildung 3.10(b):

Dieses Ziel wird dadurch sichergestellt, dass die Länge des Kraftvektors eine bestimmte maximale Länge, in Abhängigkeit der Lage der Gitterpunkte, nicht überschreitet. Die genaue Definition dieser Längen wird in diesem Kapitel erarbeitet und angegeben.

Bei der Parallelisierung des Problems administriert jeder Prozessor zwar vier bzw. acht Punkte, ist jedoch nur für die Verschiebung eines Punktes verantwortlich. Jeder Prozessor verschiebt nur seinen linken unteren und im dreidimensionalen vorderen Punkt und sendet die Informationen für diesen Punkt

an die Gebiete, die diesen Punkt administrieren, aber nicht die Berechnung der Verschiebung durchführen. Die Bestimmung der maximalen Länge der Verschiebung geschieht dabei ohne die Informationen der Verschiebung der anderen Gitterpunkte im aktuellen Zeitschritt. Daher lässt sich die Verschiebung der Gitterpunkte und die Verwaltung der Gebiete gut parallelisieren. Die genaue Berechnung der maximalen Längen wird im Folgenden erklärt.
Die Punkte $\vec{p}$, $\vec{q}$, $\vec{r}$ und $\vec{s}$ bilden, wie in Abbildung 3.11 zu sehen, die Eckpunkte eines beliebig ausgewählten Gebietes. Exemplarisch für die maximale Verschiebungslänge betrachten wir die Verschiebung des Punktes $\vec{p}$. Wie bereits erwähnt soll sicher gestellt werden, dass der Punkt $\vec{p}$ nicht auf die Diagonale, die Verbindungsgerade zwischen $\vec{q}$ und $\vec{r}$, läuft oder diese gar überschreitet. Wenn nun eine maximale Länge l_{max} definiert werden soll, mit der der Punkt $\vec{p}$ verschoben werden darf, muss man den ungünstigsten Fall betrachten. Dieser tritt genau dann ein, wenn an dem Punkt $\vec{p}$ eine Kraft senkrecht zur Diagonalen in Richtung dieser Diagonalen wirkt. Gleichzeitig werden die Punkte $\vec{q}$ und $\vec{r}$ ebenfalls mit einer Kraft senkrecht zur Diagonalen, jetzt aber in Richtung des Punktes $\vec{p}$, verschoben. Als neue Diagonale würde im nächsten Schritt die gestrichelte Linie in Abbildung 3.11 entstehen. Um in diesem nächsten Schritt sicherzustellen, dass der Punkt $\vec{p}$ nicht die gestrichelte Diagonale durchläuft, kann man eine maximale Verschiebungslänge für den Vektor $\vec{p}$ definieren. Es wird davon ausgegangen, dass in dem in Abbildung 3.11 konstruierten ungünstigsten Fall die Länge der drei eingezeichneten Verschiebungsvektoren gleich ist. Somit kann die maximale Länge l_{max} im zweidimensionalen Fall unter den beschriebenen Voraussetzungen folgendermaßen bestimmt werden:
Die maximale Länge l_{max} mit der der Punkt $\vec{p}$ verschoben werden darf, beträgt genau die Hälfte des Abstandes zwischen dem Punkt $\vec{p}$ und der Diagonalen durch $\vec{q}$ und $\vec{r}$. Der Abstand von Punkt $\vec{p}$ zu der Diagonalen durch $\vec{q}$ und $\vec{r}$ berechnet sich mit der Formel:

$$d = \left| \frac{\langle \vec{p} - \vec{q}, \vec{n} \rangle}{||\vec{n}||} \right| \tag{3.13}$$

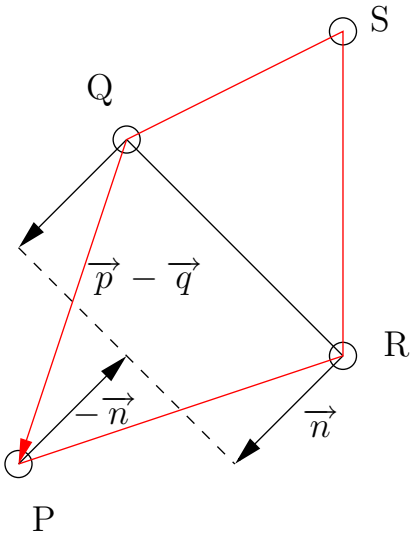


Abbildung 3.11: Einschränkung maximale Verschiebung

Der Vektor $\overrightarrow{n}$ steht senkrecht zu dem Verbindungsvektor zwischen $\overrightarrow{q}$ und $\overrightarrow{r}$, hat aber nicht zwangsläufig die Länge. Für l_{max} gilt somit: $l_{max} = \frac{d}{2}$. Die so berechnete maximale Länge für die Verschiebung des Gitterpunktes $\overrightarrow{p}$ stellt jedoch die Stabilität nur für dieses spezielle Gebiet sicher. Darüber hinaus muss man außerdem berücksichtigen, dass der Punkt $\overrightarrow{p}$ vier Nachbargebiete hat und diese Überlegung für alle vier Nachbargebiete durchgeführt werden muss. Wenn man l_{max_i} für $i \in \{1, 2, 3, 4\}$ als die maximalen Verschiebungslängen der vier angrenzenden Gebiete definiert, so muss man für die direkte maximale Verschiebunsglänge das folgende Minimum bestimmen:

$$l_{max,direkt} = \min_{i=1,2,3,4} (l_{max_i})$$

Die direkten Längen $l_{max,direkt}$, sind die Längen, welche bei der Verschiebung des aktuellen Punktes entstehen. Als indirekte Längen werden im Folgenden, die Längen bezeichnet, die durch die Verschiebung von Nachbar-Gitterpunkten berücksichtigt werden. l_{max_i} sind die maximalen Längen mit denen der Punkt $\overrightarrow{p}$ in die vier angrenzenden Gebiete verschoben werden darf. Um sicherzustellen, dass in keiner der vier angrenzenden Gebiete das Stabilitätskriterium verletzt wird, nimmt man das Minimum der vier ma-

ximalen Längen. Zusätzlich dürfen die Punkte $\vec{q}$ und $\vec{r}$ in Abbildung 3.11 nicht länger als die Länge l_{max} verschoben werden. Dies hat zur Folge, dass jeder Punkt neben den vier direkten maximalen Verschiebungslängen auch indirekte maximale Verschiebungslängen berücksichtigen muss. Der mittlere rote Gitterpunkt in Abbildung 3.12 bekommt von seinen vier Nachbargitterpunkten (in der Abbildung schwarz dargestellt) jeweils zwei indirekte maximale Längen übergeben. Für den rechten Nachbarn sind die vier Diagonalen eingezeichnet, die er nach der Verschiebung nicht überschreiten darf. Die zwei gestrichelten Linien, die außerhalb liegen haben dabei keinen Einfluss auf die maximale Verschiebungslänge des mittleren roten Punktes. Die zwei Diagonalen, die durch den roten Gitterpunkt verlaufen, sorgen dafür, dass dieser während seiner nächsten Verschiebung zusätzlich zwei indirekte maximale Längen nicht überschreiten darf. Führt man diese Überlegung für alle vier Nachbargitterpunkte durch, so erhält man acht indirekte maximale Längen, aus denen zusätzlich das Minimum ausgewählt wird.

$$l_{max,indirekt} = \min_{1 \leq i \leq 8} (l_{max_i})$$

Die endgültige Länge l_{max} wird nun bestimmt durch die minimale indirekte Länge $l_{max,indirekt}$ und durch die minimale direkte Länge $l_{max,direkt}$:

$$l_{max} = \min (l_{max,direkt}, l_{max,indirekt})$$

Wenn die Länge der Verschiebung immer unterhalb der so bestimmten Länge l_{max} liegt, hält man das Stabilitätskriterium ein und stellt sicher, dass die verwendeten Gebiete immer konvex sind. In drei Dimensionen gilt für jeden Gitterpunkt, dass er acht Diagonalebenen hat, die er nicht durchdringen darf. Die Vorgehensweise im dreidimensionale Fall geschieht, wie bereits bei anderen Problemstellungen, analog zu dem zweidimensionalen Fall mit kleinen Änderungen. Die Diagonalebene wird nun durch drei Nachbargitterpunkte gelegt. Für diese Ebene wird ein Normalenvektor $\vec{n}$ bestimmt. Dieser muss nicht notwendigerweise die Länge 1 haben. Zudem wird einer der drei Nachbargitterpunkte ausgewählt, und für die Berechnung des Abstandes verwen-

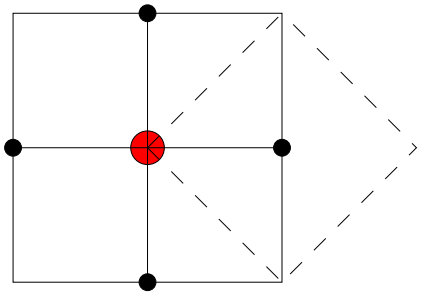


Abbildung 3.12: Indirekte maximale Längen

det. Die Abstände werden analog zu der Formel für den zweidimensionalen Fall (3.13) mittels der Hesseschen Normalenform bestimmt. Falls $\vec{r}$ der zu verschiebende Gitterpunkt ist und $\vec{q}$ einer der der Diagonalebene ist, ergibt sich für für den Abstand d der folgende Ausdruck:

$$d = \left| \frac{\langle \vec{r} - \vec{q}, \vec{n} \rangle}{||\vec{n}||} \right|$$

Die Hälfte des Abstandes zu jeder dieser Ebenen definiert wieder die direkte maximale Länge in jede Richtung.

Die Anzahl der indirekten maximalen Längen, die jeder Gitterpunkt berücksichtigen muss, verändert sich jedoch. Jeder Gitterpunkt besitzt nun sechs Nachbargitterpunkte. Jeder dieser sechs Nachbargitterpunkte sorgt für vier indirekte maximale Längen. Dies liegt daran, dass nicht wie in zweidimensionalen Fall zwei Diagonalen berücksichtigt werden sondern vier Diagonalebenen. Für $l_{max,direkt}$, $l_{max,indirekt}$ und l_{max} ergibt sich für den dreidimensionalen Fall:

$$l_{max,direkt} = \min_{1 \leq i \leq 8} (l_{max,dir_i})$$

$$l_{max,indirekt} = \min_{1 \leq i \leq 24} (l_{max,indir_i})$$

$$l_{max} = \min (l_{max,direkt}, l_{max,indirekt})$$

Anschaulich gesprochen wird sichergestellt, dass ein Gitterpunkt keine seiner gegenüberliegenden Diagonalebene durchdringt. Daraus folgt jedoch, anders

als im zweidimensionalen Fall, nicht, dass das Gebiet konvex ist. Dies liegt daran, dass ein Gebiet an jeder seiner acht Seiten wie in Kapitel 3.2.3 beschrieben, durch zwei Dreiecke begrenzt sind und diese keineswegs immer konvex sind. Bezogen auf die Abbildung 3.7(a) bedeutet dies: Das Gebiet hält die Bedingung ein, dass keine diagonalen Ebenen durchdrungen worden sind, ist aber trotzdem nicht konvex. Dies liegt daran, dass die Begrenzung des Gebietes nach oben hin einen „Knick“ nach innen macht. Trotzdem lässt sich die Zugehörigkeit der Teilchen einwandfrei bestimmen, wie in Kapitel 3.2.3 beschrieben.

Dieses Stabilitätskriterium stellt sicher, dass für den zweidimensionalen Fall alle Gebiete konvex bleiben und schließt aus, dass ein Punkt im nächsten Zeitschritt die gegenüberliegende Diagonale überschreitet. Dieses „Nichtüberschreiten“ der Diagonalen bzw. Diagonalebenen gilt sowohl für zwei Dimensionen als auch für drei Dimensionen. Dieses Kriterium zur Stabilität lässt sich im parallelen Fall gut anwenden, da man die Verschiebung der Nachbar-Gitterpunkte im aktuellen Schritt nicht benötigt. Diese Information wird durch den in diesem Kapitel beschriebenen ungünstigsten Fall nicht mehr gebraucht, da hier von der ungünstigsten Verschiebung der anderen Gitterpunkte ausgegangen wird. D.h. man kann die schlechtest mögliche Verschiebung der Nachbarn abschätzen und ist nicht auf die tatsächliche Verschiebung angewiesen. Wäre man auf die Informationen der Nachbargitterpunkte im aktuellen Schritt angewiesen, so müsste man die Gitterpunkte nacheinander verschieben und die Informationen dann jeweils weitergeben. Dies wiederum hätte eine Sequentialisierung zur Folge und könnte parallel nicht performant implementiert werden.

Bei dem Test auf Gebietszugehörigkeit mittels Vektorrechnung in zwei Dimensionen bilden konvexe Gebiete die Grundlage. Wie bereits gesehen wird bei der Gebietszugehörigkeit mittels Vektorrechnung mithilfe von Normalenvektoren getestet, ob ein Teilchen außerhalb oder innerhalb des betrachteten Gebietes liegt. Wäre das Gebiet, wie in Abbildung 3.13 zu sehen, konkav, so könnte es passieren, dass ein Teilchen nach den bisher beschriebenen Me-

thoden als außerhalb des Gebietes gesehen wird, obwohl es eigentlich in dem Gebiet liegt. Wenn das eingezeichnete Teilchen darauf hin getestet wird, ob es innerhalb oder außerhalb des Gebietes liegt, dann wird es mit der bisherigen Vorgehensweise als außerhalbliegend erkannt. Das liegt daran, dass für die Gerade mit dem farbig eingezeichneten Normalenvektor, das Teilchen fälschlicherweise als nicht innerhalb markiert werden würde. Dieser Fall kann nur bei konkaven Gebieten auftreten, da nur dann die gedachte Verlängerung von Seitenkanten (hier als gestrichelte Linie eingezeichnet) innerhalb des Gebietes liegt. Bei konvexen Gebieten kann dieser Fall nicht auftreten.

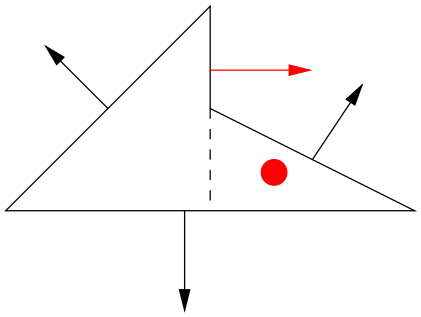


Abbildung 3.13: Fehlerhafte Zuordnung eines Teilchens

Um den Fall eines entarteten Vierecks, d.h. ein Eckpunkt liegt auf der Flächendiagonalen, zu vermeiden können abstoßende Kräfte von den gegenüberliegenden Diagonalen eingeführt werden.

In Abbildung 3.14 repräsentiert der Punkt P den zu verschiebenden Gitterpunkt und die Gerade g die gegenüberliegende Diagonale. Zur Einhaltung der obigen Bedingung erfährt der Punkt P eine abstoßende Kraft von der Gerade g. Dazu wird zunächst der Einheitsnormalenvektor $\overrightarrow{n^0}$ so bestimmt, dass er von der Gerade g nicht in Richtung des Punktes P zeigt. Da die abstoßende Kraft maßgeblich vom Abstand des Punktes zur Geraden bestimmt wird, wird im nächsten Schritt der Abstand r_1 mittels des Skalarproduktes berechnet. Die Punkte Q und R sind zwei Nachbargitterpunkte, durch die die Gerade g verläuft. Zur Bestimmung des Abstandes r_1 wird der zu dem Punkt Q gehörige Ortsvektor $\overrightarrow{q}$ verwendet.

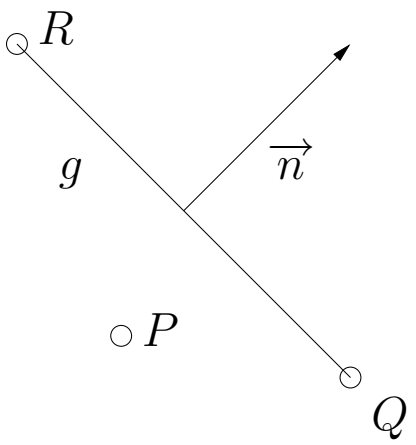


Abbildung 3.14: Berechnung der Kraft

$$r_1 = \langle (\vec{q} - \vec{p}), \vec{n} \rangle$$

Die abstoßende Kraft $\vec{f}_{ab}$ ist antiproportional zu dem Abstand r_1 und wird folgendermaßen definiert:

$$\vec{f}_{ab} = -\frac{\alpha}{|r_1|^n} \cdot \vec{n}^0$$

Die Parameter α und n sind dabei frei wählbare Parameter, die jedoch an das System und an die Systemgröße angepasst werden müssen. Ist man an einer abstoßenden Kraft interessiert, die über eine größere Distanz wirkt, so wählt man den Parameter n sehr klein (O(1)). Für sehr kurzreichweitige Kräfte wählt man ihn größer (O(10)).
Wenn $\vec{f}$ der Vektor der Kraft ist, der von dem zu verschiebenden Gitterpunkt aus ohne Berücksichtigung der abstoßenden Kräfte in Richtung der Diagonalen zeigt, so lässt sich eine untere Schranke für die Länge von $\vec{f}_{ab}$ angeben. Diese lautet wie folgt:

$$\left|\left|\vec{f}_{ab}\right|\right| > \left|\left\langle \vec{f}, \vec{n}^0 \right\rangle\right| \tag{3.14}$$

In der Ungleichung (3.14) ist $\vec{n}^0$ der Senkrechtenvektor der Diagonalen mit der Länge 1. Die rechte Seite der Ungleichung beschreibt die Länge des Vek-

tors, der entsteht, wenn man $\vec{f}$ auf den Normalenvektor $\vec{n}$ pojiziert. Wenn man nun sicherstellt, dass der Betrag der abstoßenden Kraft $\vec{f}_{ab}$ größer ist, als die Länge des projizierten Vektors, so kann der Gitterpunkt die betrachtete Diagonale nicht überschreiten. Über diese Bedingung erhält man eine Einschränkung für den Wert α, die in dieser Arbeit aber nicht weiter berücksichtigt wird.

3.4 Das Hierarchische Modell

Bei den bis hier beschriebenen Vorgehensweisen zur Gebietsaufteilung, kann in der Regel eine gute Aufteilung der Arbeit auf die Gebiete erreicht werden. Dabei stellt sich jedoch die Frage, ob die Konvergenzgeschwindigkeit, mit der diese bestmögliche Aufteilung erreicht wird, noch verbessert werden kann. Insbesondere bei sehr inhomogenen Verteilungen und sehr vielen Prozessoren, kann es lange dauern, bis sich die Prozessorgrenzen so verschoben haben, dass die Gebiete aller Prozessoren ungefähr gleich viele Teilchen enthalten.
Ein Extremfall besteht beispielsweise darin, dass ein Prozessor alle Teilchen besitzt. Hierbei würde es eine große Anzahl an Iterationsschritten dauern, bis alle weiteren Gebiete in dieses mit Teilchen besetzte Gebiet „herein gezogen würden“. Um die Konvergenz zu beschleunigen wird in diesem Kapitel die Funktionsweise des hierarchischen Modells betrachtet. Dieses hierarchische Modell beginnt immer mit einer Aufteilung in zwei Gebiete in jeder Richtung. Für diese Aufteilung wird das bereits beschriebene Loadbalancing durchgeführt. Nach Beendigung dieses Loadbalancings wird nun jedes Gebiet in 2^d möglichst gleichgroße Gebiete unterteilt. Für jedes dieser neuen Gebiete ist nun ein eigener Prozessor verantwortlich. Wenn auch für diese Gebietsaufteilung die Lastbalance durchgeführt wurde, werden die Gebiete im nächsten Schritt wieder unterteilt. Diese Unterteilung ist so lange möglich, bis die maximale Anzahl der Gebiete bzw. Prozessoren erreicht wird.
Wie die Gebiete aufgeteilt werden, ist dabei davon abhängig, ob man sich im zweidimensionalen und im dreidimensionalen Fall bewegt.

Das hierarchische Modell erfüllt somit die Aufgabe eines Präkonditionierers. D.h. die Gebiete werden nur im ersten Schritt äquidistant aufgeteilt. In weiteren Schritten sind die Gebietsgrenzen bereits präkonditioniert, da sie an das System angepasst wurden.

3.4.1 Der zweidimensionale Fall

Bei dem hierarchischen Modell in zwei Dimensionen wird immer mit einer Aufteilung in vier Gebiete bzw. Prozessoren begonnen. Die Gittergrenzen werden so verschoben wie in den vorrangegangenen Kapiteln beschrieben. Nachdem dieser Teil konvergiert ist (hier muss er nicht unbedingt völlig auskonverigiert sein) wird jedes der vier Gebiete in vier weitere Gebiete aufgeteilt. Diese Aufteilung geschieht über die folgenden Schritte:

- Bekannt sind die vier Eckpunkte ($\vec{e}_1,\ \vec{e}_2,\ \vec{e}_3,\ \vec{e}_4$) jedes Prozessorgebietes. Über diese vier Eckpunkte lassen sich die Mittelpunkte aller begrenzenden Kanten, über das arithmetische Mittel des x-Wertes und des y-Wertes bestimmen:

$$\vec{m}_i = \frac{1}{2} \cdot (\vec{e}_i + \vec{e}_{i+1}), i \in \{1, 2, 3\}$$

$$\vec{m}_4 = \frac{1}{2} \cdot (\vec{e}_4 + \vec{e}_1)$$

  Nach dieser Formel bezeichnen $\vec{m}_1$ bis $\vec{m}_4$ die Mittelpunkte der begrenzenden Seitenlinien eines Gebietes.

- Im nächsten Schritt werden die jeweils gegenüberliegenden Seitenmittelpunkte miteinander verbunden. Der Schnittpunkt dieser beiden Verbindungsgeraden liegt wieder im betrachteten Gebiet und wird als Mittelpunkt $\vec{m}_5$ bezeichnet.

- Das ursprüngliche Gebiet mit den Eckpunkten $\vec{e}_1$, $\vec{e}_2$, $\vec{e}_3$, $\vec{e}_4$ erfährt nun die Aufteilung, wie in Abbildung 3.15 zu erkennen ist.

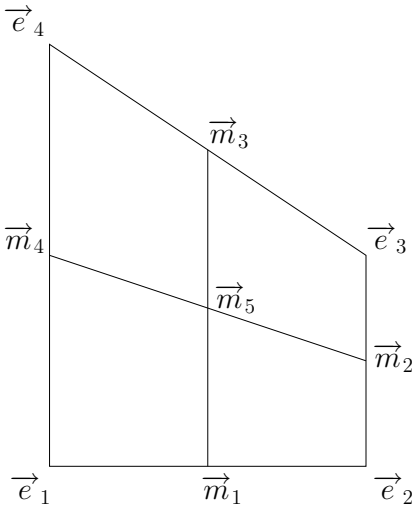


Abbildung 3.15: Neue Aufteilung des Gebiets

Jedes der vier neuen Gebiete wird nun einem Prozessor zugeordnet. Der Prozessor, der vorher verantwortlich für das gesamte Gebiet war, verteilt nun alle seine Teilchen auf seine „Unterprozessoren“. Dabei bleibt er jedoch für eines der vier neuen Teilgebiete verantwortlich.
Im nächsten Schritt ist jeder der vier „Unterprozessoren“ der verantwortliche Prozessor in seinem Gebiet und führt von diesem erneut eine Aufteilung in vier Untergebiete, wie oben beschrieben, durch. Diese wiederholte Aufteilung der Gebiete kann so lange durchgeführt werden, bis die gewünschte Anzahl an Prozessoren erreicht wird. Hier ist zu beachten, dass sich im zweidimensionalen Fall die Anzahl der Gebiete in jedem Schritt vervierfacht.
Die Ergebnisse für das hierarchische Modell, insbesondere im Vergleich zu dem nicht hierarchischen Modell, sind Kapitel 4.4 zu entnehmen.

3.4.2 Der dreidimensionale Fall

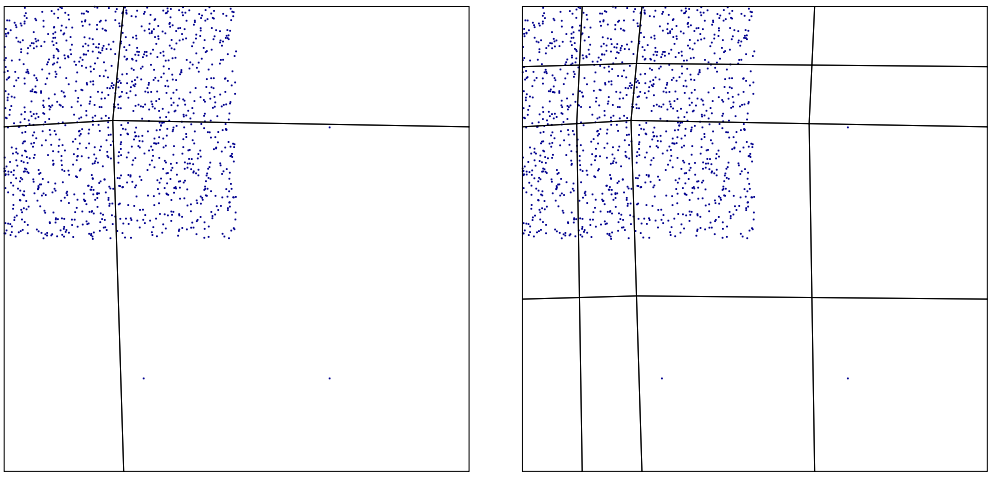
Auch im hierarchischen Modell ist der dreidimensionale Fall das Analogon zu dem bereits gesehenen zweidimensionalen Fall. Im Wesentlichen gibt es jedoch zwei Unterschiede. Der erste Unterschied besteht lediglich darin, dass ein Gebiet nicht mehr in vier sondern in acht neue Teilgebiete aufgeteilt wird.

Der zweite etwas aufwändigere Unterschied geht der Frage nach, wie diese acht Gebiete aus dem einen ursprünglichen Gebiet gebildet werden. Wenn wir ein ursprüngliches Gebiet betrachten, so besteht dieses aus acht Eckpunkten und zwölf begrenzenden Ebenen. Bei der Betrachtung der oberen Grenze des Gebietes, welche durch die vier oberen Randpunkte eindeutig gegeben ist, ist zu berücksichtigen, dass diese Grenze sich in zwei Teilflächen unterteilt. D.h., dass in der Regel weder alle vier Punkte, noch die diagonalen Verbindungsgeraden in einer Ebene liegen. Lediglich die Diagonale zwischen den Punkten, die am „Knick“ der oberen begrenzenden Fläche liegen, verläuft in beiden begrenzenden Flächen. In der Abbildung 3.7 bilden die Punkte $\vec{p}_1$, $\vec{p}_2$ und $\vec{p}_3$ eine Ebene. Die Punkte $\vec{p}_2$ und $\vec{p}_3$ und $\vec{p}_4$ bilden die zweite nach oben begrenzende Ebene. In dieser Abbildung ist zu erkennen, dass die Verbindungsgerade zwischen $\vec{p}_2$ und $\vec{p}_3$ in beiden Ebenen enthalten ist.

Für die Bestimmung der acht neuen Gebiete werden nun die Mittelpunkte der vier Seitenkanten bestimmt sowie der Mittelpunkt der Geraden durch $\vec{p}_2$ und $\vec{p}_3$. Diese Mittelpunkte werden für alle sechs Seitenflächen bzw. zwölf begrenzenden Ebenen ermittelt. Außerdem wird der Massenschwerpunkt aller acht Eckpunkte des alten Gebietes bestimmt. Da es nicht sein kann, dass ein Gitterpunkt eine seiner angrenzenden Diagonalflächen durchdringt, ist sichergestellt, dass der Massenschwerpunkt immer innerhalb des Gebietes liegt. Nun bilden die Verbindungsgeraden von allen berechneten Mittelpunkten zu dem Massenschwerpunkt die Kanten der neuen begrenzenden Gebiete. Diese Mittelpunkte werden nun als Eckpunkte der neuen Gebiete zuzüglich zu den bereits vorhandenen Eckpunkten den neuen Gebieten zugeordnet.

3.4.3 Hierarchisches Modell mit Reibung

Wenn man das hierarchische Modell so anwendet, wie bis hier her beschrieben, so erfüllt es die Aufgabe eines Präkonditionierers. D.h. in jeder Hierarchiestufe startet man mit einem bestimmten Gitter, welches aber nicht äquidistant ist, sondern durch die bereits berechneten Hierarchiestufen vor-



(a) Ende der 1. Hierarchiestufe (b) Beginn der 2. Hierarchiestufe

Abbildung 3.16: Übergang von der 1. Hierarchiestufe zur 2. Hierarchiestufe

konditioniert ist. Hierbei werden die Gitterpunkte, welche bereits in höheren Hierarchiestufen verschoben wurden, genauso behandelt, wie alle weiteren Gitterpunkte. Dies kann unter Umständen einen unerwünschten Nebeneffekt zur Folge haben. Man kann sich Beispiele konstruieren, in denen ein Gitterpunkt bereits einen guten Ort für die ausgeglichene Lastbalance gefunden hat. Im nächsten Hierarchieschritt kann es jedoch passieren, dass dieser Gitterpunkt aus dieser bereits sehr guten Lage herausgezogen wird. Ein Beispiel für diese Überlegung zeigt die Abbildung 3.16. Der Mittlere Gitterpunkt ist nach dem ersten Hierarchieschritt bereits recht gut positioniert. Zu Beginn des zweiten Hierarchischrittes wird er jedoch aus dieser Position heraus in die eigentlich falsche Richtung, nach links oben, gezogen. Dies liegt daran, dass die Teilchenanzahl des Gebietes, welches rechts unten von dem betrachteten Gitterpunkt liegt, größer ist, als die Anzahl der Teilchen des Gebietes, welches unmittelbar links oben angrenzt. Um solche ungewollten Effekte zu vermeiden gibt es die Möglichkeit den Gitterpunkten eine Reibung mitzugeben.

Diese Reibung äußert sich anschaulich, darin, dass Gitterpunkte, die in höheren Hierarchieschritten bereits verschoben wurden mit einer schwereren

Masse bzw. einer Reibung belegt werden. Dies hat zur Folge, dass diese Gitterpunkte sich wesentlich weniger oder in bestimmten Fällen gar nicht mehr bewegen. Somit könnte man erreichen, dass bereits gut konvergierte Gitterpunkte ihre Position nicht mehr „verschlechtern“ können. Es gibt die Möglichkeit Gitterpunkte aus höheren Hierarchieschritten gar nicht mehr zu verschieben. Dies würde dann Sinn ergeben, wenn man sicher sein könnte, dass sie bereits optimal auskonvergiert sind. Eine zweite Möglichkeit ist, dass man Gitterpunkte nach ihren Hierarchiestufen gewichtet. Gitterpunkte die im aktuellen Hierarchieschritt erzeugt wurden, erfahren die volle Kraft und Verschiebung. Gitterpunkte, die bereits in vorangegangenen Schritten betrachtet wurden, werden nicht mit voller Kraft verschoben. Sie werden nun nur noch mit dem 4^i-ten Teil ihrer eigentlichen Kraft verschoben, wenn man davon ausgeht, dass sie bereits in den letzten i Hierarchieschritten verschoben wurden. In den Abbildungen 3.17, 3.18 und 3.19 werden die Zusammenhänge zwischen den Matrizen, die die Anteile der Verschiebungen für jeden Gitterpunkt beinhalten, und den Gitterpunkten im System veranschaulicht. Die Matrix zur Abbildung 3.17 besagt, dass im ersten Hierarchieschritt alle Gitterpunkte die volle Kraft erfahren. Ausgenommen davon sind die vier Eckpunkte, die auch in den weiteren Hierarchieschritten keine Kraft erfahren. Die neu eingefügten Gitterpunkte und Gittergrenzen sind in der Abbildung 3.18 rot dargestellt. Die schwarzen Punkte, die bereits in Abbildung 3.17 vorhanden waren, werden in der Matrix 3.18 nur noch mit dem Anteil $\frac{1}{4}$ gewichtet. Die roten Punkte erfahren die volle Kraft und werden in der Matrix daher jeweils durch eine 1 repräsentiert. Im dritten Hierarchieschritt werden die neuen Gittergrenzen in Abbildung 3.19 grün dargestellt. Die schwarzen Gitterpunkte erfahren nur noch $\frac{1}{16}$ der Kraft, die roten Gitterpunkte erfahren in diesem Schritt nur noch $\frac{1}{4}$ der eigentlichen Kraft. Lediglich die „neuen“ grün eingezeichneten Gitterpunkte werden mit 100% ihrer Kraft verschoben.

Die in diesem Abschnitt beschriebene Vorgehensweise für die Verschiebung der Gitterpunkte mit Reibung ist im dreidimensionalen gleich anzuwenden. Der einzige Unterschied zu den bereits betrachteten Problemen, darin, dass

$$A^{(0)} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

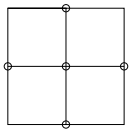


Abbildung 3.17: Erster Hierarchieschritt

$$A^{(1)} = \begin{pmatrix} 0 & 1 & \frac{1}{4} & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ \frac{1}{4} & 1 & \frac{1}{4} & 1 & \frac{1}{4} \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & \frac{1}{4} & 1 & 0 \end{pmatrix}$$

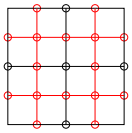


Abbildung 3.18: Zweiter Hierarchieschritt

$$A^{(2)} = \begin{pmatrix} 0 & 1 & \frac{1}{4} & 1 & \frac{1}{16} & 1 & \frac{1}{4} & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \frac{1}{4} & 1 & \frac{1}{4} & 1 & \frac{1}{4} & 1 & \frac{1}{4} & 1 & \frac{1}{4} \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \frac{1}{16} & 1 & \frac{1}{4} & 1 & \frac{1}{16} & 1 & \frac{1}{4} & 1 & \frac{1}{16} \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \frac{1}{4} & 1 & \frac{1}{4} & 1 & \frac{1}{4} & 1 & \frac{1}{4} & 1 & \frac{1}{4} \\ 0 & 1 & \frac{1}{4} & 1 & \frac{1}{16} & 1 & \frac{1}{4} & 1 & 0 \end{pmatrix}$$

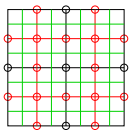


Abbildung 3.19: Dritter Hierarchieschritt

man mit einer dreidimensionalen Matrix zu arbeiten ist. Außerdem ist leicht zu erkennen, dass die Gitterpunkte, welche bereits im 1. Hierarchieschritt existierten mit dem $\frac{1}{4^k}$-ten Anteil ihrer eigentlichen Kraft verschoben werden. Dies gilt ebenfalls für $k \geq 3$. Alternativ zu den hier beschriebenen Reibungen wäre es auch möglich, die Matrixeinträge, die ungleich 1 sind direkt auf Null zu setzten. In diesem Fall würden bereits veränderte Gitterpunkte im aktuellen Schritt gar keine Kraft mehr erfahren und würden ihren Platz bis zum Ende der Präkonditionierung nicht mehr verändern. Die Gebiete in den Abbildungen 3.17, 3.18 und 3.19 sind äquidistant dargestellt, um den direkten Bezug zu den verwendeten Matrizen herzustellen. In der Anwendung werden diese Gitter in allen Schritten verzerrt sein und von der äquidistanten Form abweichen. Die zugrunde liegenden Überlegungen zu der Reibung mit der die Gitterpunkte versehen werden, spielt dies jedoch keine Rolle.

3.5 Mathematisches Modell

Nachdem das Verfahren zur Arbeitsverteilung erklärt wurde, wird in diesem Kapitel das zugrundeliegende mathematische Modell erläutert. Dabei wird auf die Arbeitsverteilung und den Arbeitsausgleich sowie auf eine Abschätzung für eine gegebene Fehlerschranke eingegangen.

3.5.1 Arbeitsausgleich durch Übertragung von lokalen Arbeitsdifferenzen

In diesem Kapitel wird geprüft, ob die globale gleichmäßige Verteilung der Arbeiten auf die verschiedenen Prozessoren mittels dem lokalen Austausch von Arbeitsdifferenzen möglich ist.

Dies wird für den dreidimensionalen Fall gezeigt. Die Abbildung 3.20 verdeutlicht an einem zweidimensionalen Beispiel die Vorgehensweise. In dieser Abbildung wird ein System der Form 3×3 betrachtet. Die betrachteten Ar-

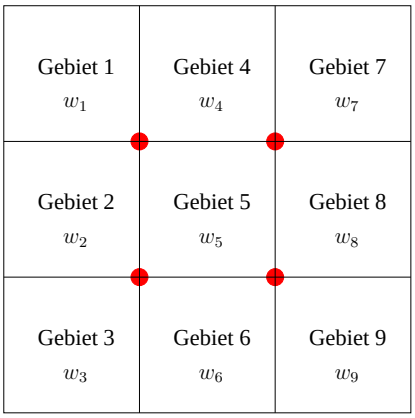


Abbildung 3.20: Zweidimensionale Nummerierung der Gebiete

beiten in jedem der neun Gebiete werden mit

$$w_i, i \in \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

bezeichnet. $\frac{1}{\gamma}$ ist der Anteil der Arbeitsdifferenz der zwischen zwei Gebieten übertragen wird, dabei gilt $\gamma \in \mathbb{R}^+$. Es wird davon ausgegangen, dass zur Übertragung der Arbeiten nur innere Punkte bewegt werden. Für die Arbeitsübertragung zwischen Gebiet 1 und Gebiet 2 gilt somit:

$$\Delta w_{1,2}^{(k)} = \frac{w_1^{(k)} - w_2^{(k)}}{\gamma} \tag{3.15}$$

Das Vorzeichen dieses Ausdrucks hat die Aufgabe sicher zu stellen, dass die Arbeit immer von dem Gebiet mit größerer Arbeit in das Gebiet mit der kleineren Arbeit übertragen wird. Falls der Ausdruck (3.15) negativ ist, so wird Arbeit vom Nachbargebiet empfangen, hier würde dies bedeuten, dass das Gebiet 1 von Gebiet 2 Arbeit empfangen würde. Ist der Ausdruck in der Gl. (3.15) positiv, dann wird Arbeit von Gebiet 1 nach Gebiet 2 abgegeben. Desweiteren muss man berücksichtigen, dass das Gebiet 1 zusätzlich mit den Gebieten 4 und 5 gemeinsame innere Gitterpunkte besitzt, alle inneren Punkte sind in der Abbildung 3.20 rot dargestellt. Das Gebiet 2 hat beispielsweise mit dem Gebiet 5 zwei innere Punkte gemeinsam. D.h., dass der Arbeitsaus-

gleich zwischen diesen beiden Gebieten doppelt durchgeführt werden muss. Im dreidimensionalen Fall können zwei Gebiete außerdem noch vier gemeinsame innere Punkte haben, so dass der Arbeitsausgleich zwischen diesen beiden Gebieten vierfach gewertet wird. Für die Arbeit im Schritt $k+1$ ergibt sich somit:

$$\begin{aligned} w_i^{(k+1)} &= w_i^{(k)} - \sum_{j \in B} \left(\alpha_j \frac{w_i^{(k)} - w_j^{(k)}}{\gamma} \right) \\ &= \frac{\gamma - \sum_{k=1}^{n} \alpha_k}{\gamma} w_i^{(k)} + \sum_{j \in B} \left(\frac{1}{\gamma} w_j^{(k)} \right) \end{aligned} \tag{3.16}$$

Hierbei ist B die Menge, die die Indizes aller Nachbargebiete des Gebietes i beinhaltet und α_j beschreibt die Anzahl der gemeinsamen inneren Punkte zwischen dem Gebiet i und dem Gebiet j. Die Gl. (3.16) ermöglicht darüber hinaus die Schreibweise als Matrix Vektorprodukt, so dass die Arbeitsverteilung im Schritt $k+1$ über die Gl. (3.17) formuliert werden kann. Dies gilt für den zwei- und dreidimensionalen Fall.

$$\vec{w}^{(k+1)} = A_\omega(\gamma) \cdot \vec{w}^{(k)} \tag{3.17}$$

mit

$$\vec{w}^{(k)} = \begin{pmatrix} w_1^{(k)} \\ w_2^{(k)} \\ \vdots \\ w_n^{(k)} \end{pmatrix} \tag{3.18}$$

Für das Beispiel aus Abbildung 3.20 kann man für den zweidimensionalen Fall die Matrix $A_\omega(\gamma)$ folgendermaßen formulieren:

Die Matrix A_1 ist zeilenweise zu lesen. Nach Gl. (3.16) liefert die erste Zeile der Matrix multipliziert mit dem Vektor $\vec{w}^{(k)}$ aus Gl. (3.18) die Arbeit im nächsten Schritt für das erste Gebiet. Wie bereits beschrieben tauscht Gebiet 1 mit den Gebieten $2, 4$ und 5 über einen Gitterpunkt Arbeit aus. Dies ist

$$A_1 := \frac{1}{\gamma} \left(\begin{array}{ccc:ccc:ccc} \gamma-3 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & \gamma-6 & 1 & 1 & 2 & 1 & 0 & 0 & 0 \\ 0 & 1 & \gamma-3 & 0 & 1 & 1 & 0 & 0 & 0 \\ \hdashline 1 & 1 & 0 & \gamma-6 & 2 & 0 & 1 & 1 & 0 \\ 1 & 2 & 1 & 2 & \gamma-12 & 2 & 1 & 2 & 1 \\ 0 & 1 & 1 & 0 & 2 & \gamma-6 & 0 & 1 & 1 \\ \hdashline 0 & 0 & 0 & 1 & 1 & 0 & \gamma-3 & 1 & 0 \\ 0 & 0 & 0 & 1 & 2 & 1 & 1 & \gamma-6 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & \gamma-3 \end{array}\right) \in \mathbb{R}^{9\times 9}$$

in der Matrix dadurch zu erkennen, dass in der ersten Zeile an genau diesen Stellen den Wert 1 steht. Für die weiteren Zeilen gilt die gleiche Vorgehensweise. Das mittlere Gebiet, das Gebiet 5 aus Abbildung 3.20, tauscht mit allen Gebieten Arbeit aus, daher besitzt in dieser Zeile jede Stelle einen Wert größer als Null. Die ersten drei Zeilen der Matrix beschreiben die Arbeitsverteilung der Gebiete der ersten Spalte der Abbildung 3.20. Die Zeilen $4, 5$ und 6 geben die Arbeitsverteilung für die mittlere Spalte an und die drei restlichen Zeilen für die rechte Spalte des Systems. Darüberhinaus lässt sich die Struktur der Matrix mithilfe der gestrichelten Linien in Gl. (3.5.1) erkennen. Diese Linien unterteilen die Matrix in Matrizen der Form 3×3. Wenn man das physikalische System in y-Richtung in n Gebiete aufteilen würde, so hätten diese Untermatrizen die Form $n \times n$. Zudem ist zu erkennen, dass drei dieser Blockmatrizen untereinander stehen. Diese Anzahl ist gegeben durch die Anzahl der Gebiete des physikalischen Systems in x-Richtung. Die hier beschriebene Herangehensweise für das zweidimensionale Beispiel bietet die Grundlage für den nun folgenden dreidimensionalen Fall.

Die Matrix $A_\omega(\gamma)$ wird nun allgemein für drei Dimensionen definiert. Aus der Gl. (3.16) kann man für ein dreidimensionales System der Größe $(m \times n \times k)$ mit den folgenden Matrizen ein Matrix-System der Größe $(m \cdot n \cdot k) \times (m \cdot n \cdot k)$ erstellen, die genau der Matrix $A_\omega(\gamma)$ entspricht.

Dazu wird jedes Gebiet über ein eindeutiges Zahlentrippel definiert. Das Zahlentrippel besitzt dabei die folgende Gestalt:

$$(x, y, z), \quad x \in \{1, ..., m\},\ y \in \{1, ..., n\},\ z \in \{1, ..., k\}$$

$$A_1 := \frac{1}{\gamma} \begin{pmatrix} \gamma-7 & 1 & 0 & \cdots & \cdots & 0 \\ 1 & \gamma-14 & 1 & 0 & \cdots & \vdots \\ 0 & 1 & \gamma-14 & 1 & \cdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & 0 & 1 & \gamma-14 & 1 \\ 0 & \cdots & \cdots & \cdots & 1 & \gamma-7 \end{pmatrix} \in \mathbb{R}^{n \times n}$$

Die Matrix A_1 wird durch alle Gebiete der Form $(1, y, 1)$, mit $y \in \{1, ..., n\}$, bestimmt, d.h. ein Eckpunkt tauscht in drei Dimensionen mit sieben anderen Gebieten jeweils über einen inneren Punkt Arbeit aus. Die Gebiete, welche an einer Seitenkante des Systems liegen tauschen mit elf Gebieten Arbeit aus. Mit acht dieser elf Gebieten wird die Arbeit einfach ausgetauscht und mit drei Gebieten doppelt. Somit kommt es insgesamt zu 14 Übertragungen von lokalen Arbeitsdifferenzen. Die im Folgenden definiert Matrix A_2 wird über alle Gebiete der folgenden Form festgelegt:

$$(x, y, 1)\,, \quad x \in \{2, ..., m-1\},\ y \in \{1, ..., n\}$$

Die Anzahl der Übertragungen von lokalen Arbeitsdifferenzen kann mit der analogen Vorgehensweise wie für die Matrix A_1 bestimmt werden. Für Gebiete, die sich am Rand befinden werden nun 14 übertragene Arbeitsdifferenzen betrachtet und für alle weiteren Gebiete sind es 28 Arbeitsdifferenzen. Somit ergibt sich für die Matrix A_2:

$$A_2 := \frac{1}{\gamma} \begin{pmatrix} \gamma-14 & 2 & 0 & \cdots & \cdots & 0 \\ 2 & \gamma-28 & 2 & 0 & \cdots & \vdots \\ 0 & 2 & \gamma-28 & 2 & \cdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & 0 & 2 & \gamma-28 & 2 \\ 0 & \cdots & \cdots & \cdots & 2 & \gamma-14 \end{pmatrix} \in \mathbb{R}^{n \times n}$$

Für alle Gebiete, welche die nächste Bedingung erfüllen kann man durch

analoge Betrachtung die Matrix A_3 definieren.

$$(x, y, z)\,, \quad x \in \{2, ..., m-1\},\ y \in \{1, ..., n\},\ z \in \{2, ..., k-1\}$$

Dabei werden bei jedem inneren Gebiet 56 Arbeitsdifferenzen übertragen.

$$A_3 := \frac{1}{\gamma} \begin{pmatrix} \gamma - 28 & 4 & 0 & \cdots & \cdots & 0 \\ 4 & \gamma - 56 & 4 & 0 & \cdots & \vdots \\ 0 & 4 & \gamma - 56 & 4 & \cdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & 0 & 4 & \gamma - 56 & 4 \\ 0 & \cdots & \cdots & \cdots & 4 & \gamma - 28 \end{pmatrix} \in \mathbb{R}^{n \times n}$$

Die Matrizen B_1 und B_2 werden an den entsprechenden Stellen so eingesetzt, dass die Gl. (3.16) für alle Komponenten erfüllt ist.

$$B_1 := \frac{1}{\gamma} \begin{pmatrix} 1 & 1 & 0 & \cdots & \cdots & 0 \\ 1 & 2 & 1 & 0 & \cdots & \vdots \\ 0 & 1 & 2 & 1 & \cdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & 0 & 1 & 2 & 1 \\ 0 & \cdots & \cdots & \cdots & 1 & 1 \end{pmatrix} \in \mathbb{R}^{n \times n}$$

$$B_2 := 2 \cdot B_1$$

Im nächsten Schritt werden die Matrizen wie folgt zusammengefasst:

$$A^* := \begin{pmatrix} A_1 & B_1 & 0 & \cdots & \cdots & 0 \\ B_1 & A_2 & B_1 & 0 & \cdots & \vdots \\ 0 & B_1 & A_2 & B_1 & \cdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & 0 & B_1 & A_2 & B_1 \\ 0 & \cdots & \cdots & \cdots & B_1 & A_1 \end{pmatrix} \in \mathbb{R}^{(n \cdot m) \times (n \cdot m)}$$

$$B^* := \begin{pmatrix} A_2 & B_2 & 0 & \cdots & \cdots & 0 \\ B_2 & A_3 & B_2 & 0 & \cdots & \vdots \\ 0 & B_2 & A_3 & B_2 & \cdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & 0 & B_2 & A_3 & B_2 \\ 0 & \cdots & \cdots & \cdots & B_2 & A_2 \end{pmatrix} \in \mathbb{R}^{(n \cdot m) \times (n \cdot m)}$$

$$C^* := \begin{pmatrix} B_1 & B_1 & 0 & \cdots & \cdots & 0 \\ B_2 & B_2 & B_1 & 0 & \cdots & \vdots \\ 0 & B_1 & B_2 & B_1 & \cdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & 0 & B_1 & B_2 & B_1 \\ 0 & \cdots & \cdots & \cdots & B_1 & B_1 \end{pmatrix} \in \mathbb{R}^{(n \cdot m) \times (n \cdot m)}$$

Die Dimensionen der Matrizen A_1, A_2, A_3, B_1 und B_2 werden ausschließlich dadurch bestimmt in wie viele Gebiete das System in Richtung der ersten Dimension unterteilt ist. Die Matrizen A^*, B^* und C^* werden in ihren Dimensionen zusätzlich durch die Anzahl der Gebiete in der 2. Dimension bestimmt. Zur Konstruktion der gesamten Matrix zum Arbeitsausgleich wird zusätzlich noch die Information für die letzte Dimension verwendet. Das gesamte System setzt sich mittels der bekannten Matrizen so zusammen:

$$A_\omega(\gamma) := \begin{pmatrix} A^* & C^* & 0 & \cdots & \cdots & 0 \\ C^* & B^* & C^* & 0 & \cdots & \vdots \\ 0 & C^* & B^* & C^* & \cdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & 0 & C^* & B^* & C^* \\ 0 & \cdots & \cdots & \cdots & C^* & A^* \end{pmatrix} \in \mathbb{R}^{(n \cdot m \cdot k) \times (n \cdot m \cdot k)}$$

Die so erzeugte Matrix $A_\omega(\gamma)$ ist eine symmetrische Bandmatrix. Die Arbeitsverteilung im gesamten System ist gegeben durch den Startvektor $\vec{w}^{(0)}$. Die Matrix $A_\omega(\gamma)$ wird wiederholt auf den Startvektor angewendet. Wenn das Verfahren stabil ist, so bewirkt diese Anwendung der Matrix eine Gleich-

verteilung der Arbeit in den Gebieten und somit auch in dem Vektor $\vec{w}^{(k)}$. Zur Prüfung, ob das Iterationsverfahren tatsächlich konvergiert, werden die Eigenwerte λ_i betrachtet.
Bekannt ist, dass auf der Diagonalen der Matrix nur Werte der Form $\frac{\gamma - x_i}{\gamma}$ stehen. Die Zeilensumme aller Nichtdiagonalelemente ergibt genau den Wert x_i. Zudem kann die Matrix $A_\omega(\gamma)$ wie in Gl. (3.19) dargestellt werden. Dort ist die Matrix vereinfacht dargestellt (Nicht-Diagonalelemente sind weggelassen, Minimum und Maximum der Diagonalelemente sind gezeigt), um die wesentlichen Punkte besser erkennen zu können.

$$A_\omega(\gamma) = \mathbb{1}_{n \cdot m \cdot k} + \frac{1}{\gamma} \begin{pmatrix} -x_{i,min} & & & & \\ & \ddots & & & \\ & & -x_{i,max} & & \\ & & & \ddots & \\ & & & & -x_{i,min} \end{pmatrix} \tag{3.19}$$

Da die Eigenwerte der Matrix $A_\omega(\gamma)$ bestimmt werden sollen und bekannt ist, dass die Eigenwerte der Einheitsmatrix alle eins ergeben, müssen nun die Eigenwerte für den zweiten Summanden abgeschätzt werden. Dies geschieht mittels Gerschgorin-Kreisen [8]. Die allgemeine Definition aus Gl. (3.20) kann für die hier betrachtete Matrix ohne weiter Einschränkungen angenommen werden. Für alle Eigenwerte gilt, dass sie innerhalb der Gerschgorin-Kreise liegen. Somit gilt für alle

$$\lambda_i, i \in \{1, 2, \cdots, (m \cdot n \cdot k)\}$$

dass

$$\lambda_i \in \overline{S}_i$$

Dabei ist $\overline{S}_i$ definiert wie folgt:

$$\overline{S}_i := \overline{S} \left(a_{ii}, \sum_{i=1, i \neq j}^{n} |a_{ij}| \right) \tag{3.20}$$

wobei $\overline{S}$(x, r) mit x $\in \mathbb{C}$, r$\in \mathbb{R}^+$ den abgeschlossenen Kreis mit Radius r um den Punkt x bezeichnet.
Außerdem ist die Matrix symmetrisch, so dass alle Eigenwerte reell sind. Ziel ist es, dass für die Eigenwerte λ_i der Matrix $A_\omega(\gamma)$ gilt $|\lambda_i| < 1$, damit das Verfahren konvergiert. Als Begründung dazu siehe Gl. (3.22). Das Spektrum für die Eigenwerte der Matrix aus Gl. (3.19) liegt im Intervall $[-112, 0]$. Nun kann man eine Abschätzung für γ folgendermaßen bestimmen:

$$\begin{aligned} \left|1 - \frac{112}{\gamma}\right| &\leq 1 \\ \overset{\gamma \geq 0}{\Leftrightarrow} \qquad \gamma &\geq 56 \end{aligned} \tag{3.21}$$

Bis hier wurde gezeigt, dass für alle Eigenwerte gilt $|\lambda_i| \leq 1$. Wählt man $\gamma > 56$, so ergibt sich nach Ungleichung (3.21), dass alle Eigenwerte im Intervall $(-1, 1]$ liegen. Für die Umformungen in Gl. (3.22) benötigt man zusätzlich, dass der größte Eigenwert 1 nur einmal auftritt. Im Folgenden wird gezeigt, dass die Vielfachheit des Eigenwertes $\lambda_1 = 1$ genau eins beträgt. Dazu wird das Perron-Frobenius Theorem [11] verwendet. Das Perron-Frobenius Theorem besagt:
Ist die Matrix $A \in \mathbb{R}^{n \times n}$ irreduzibel, so folgt daraus:

1. $r = \rho(A) \in \sigma(A)$ und $r > 0$
   r ist der Spektralradius der Matrix A. In unserem Beispiel beträgt der Spektralradius 1.

2. $alg\ mult_A(r) = 1$
   Die algebraische Vielfachheit von r beträgt 1. D.h. für die betrachtete Matrix, dass der betragsgrößte Eigenwert genau einmal vorkommt. Mit der Einschränkung $\gamma > 56$ ist nur der Wert 1 für diesen Eigenwert möglich.

Auf Grund der Struktur der Matrix $A_\omega(\gamma)$ ist in Analogie zu den Matrizen der finiten Differenzen Diskretisierung bei elliptischen Differentialgleichun-

gen, die Matrix $A_\omega(\gamma)$ irreduzibel. Insgesamt wurde somit gezeigt, dass für $\gamma > 56$ der Spektralradius $r = 1$ ist und alle Eigenwerte in dem Intervall $(-1, 1]$ liegen. Nach dem Perron-Frobenius Theorem gilt, dass der Eigenwert für den Spektralradius r genau einmal vorkommt. Da $\lambda_1 = -1$ mittels der Bedingung $\gamma > 56$ ausgeschlossen wurde, wurde gezeigt, dass der Eigenwert $\lambda_1 = 1$ einfacher und betragsgrößter Eigenwert der Matrix A ist.

Die Verteilung der Arbeit nach k Iterationen steht auf dem Vektor $\vec{w}^{(k)}$. Der Startvektor $\vec{w}^{(0)}$ kann als Linearkombination der normierten Eigenvektoren der Matrix bzw. des Matrixoperators geschrieben werden, also als $\sum_{i=1}^{n} \alpha_i \vec{e}_i$. Bei wiederholter Anwendung der Matrix $A_\omega(\gamma)$ auf den Startvektor ergibt sich der folgende Ausdruck:

$$\begin{aligned} & \vec{w}^{\{l\}} \\ = \; & (A_\omega(\gamma))^l \cdot \vec{w}^{\{0\}} \\ = \; & (A_\omega(\gamma))^l \cdot \sum_{i=1,..,n} (\alpha_i \cdot \vec{e}_i) \\ = \; & \sum_{i=1,..,n} (\alpha_i \cdot \lambda_i^l \cdot e_i) \underset{l \to \infty}{\to} \alpha_1 \cdot \vec{e}_1 \end{aligned} \tag{3.22}$$

Der letzte Schritt dieser Umformung gilt nur dann, wenn λ_1 einfacher und betragsgrößter Eigenwert ist. Dies wurde bereits mithilfe des Perron-Frobenius Theorem gezeigt. Zusätzlich ist $\vec{e}_1$ der zu λ_1 gehörige Eigenvektor. Wird nun der Startvektor auf den normierten Eigenvektor projiziert, dann gilt:

$$\begin{aligned} & \alpha_1 \cdot \vec{e}_1 \\ = \; & \left\langle \vec{e}_1, \vec{w}^{\{0\}} \right\rangle \cdot \vec{e}_1 \\ = \; & \frac{1}{\sqrt{n}} \cdot \sum_{i=1}^{n} w_i^{\{0\}} \cdot \frac{1}{\sqrt{n}} (1, \ldots, 1)^T \\ = \; & \frac{1}{n} \cdot \sum_{i=1}^{n} w_i^{\{0\}} \cdot (1, \ldots, 1)^T \end{aligned} \tag{3.23}$$

Im Grenzwert $l \to \infty$ konvergiert der Startvektor der Arbeit komponentenweise gegen den globalen Mittelwert der Arbeiten. Auf diese Weise wurde

gezeigt, dass durch die Übertragung von Arbeitsanteilen der Algorithmus gegen den globalen Ausgleich der Arbeiten konvergiert.

3.5.2 Bedingungen an die Verteilung der Arbeit und der Dichte

Zur Veranschaulichung wird an dieser Stelle ein zweidimensionales System betrachtet, welches in vier Gebiete unterteilt ist. Dabei beschreibt $W_i^{(k)}$ die Arbeit im i-ten Gebiet und k-ten Iterationsschritt. In der Ungleichung (3.24) beschreibt $W_j^{(k)}$ die Arbeit des Gebietes in dem im k-ten Iterationsschritt die meiste Arbeit vorhanden ist. In der anschließenden Summe werden die Arbeiten der restlichen drei Gebiete aufsummiert. Wenn die Ungleichung (3.24) erfüllt ist, gilt, dass die Arbeitsdifferenz zwischen dem Gebiet mit der größten Arbeit und dem arithmetischen Mittel der restlichen Gebiete in jedem Iterationsschritt abnimmt. Im Folgenden wird untersucht, für welche Voraussetzungen die Ungleichung eingehalten wird.

$$\left| W_j^{(k+1)} - \frac{1}{3} \sum_{i=1}^{3} W_i^{(k+1)} \right| < \left| W_j^{(k)} - \frac{1}{3} \sum_{i=1}^{3} W_i^{(k)} \right| \tag{3.24}$$

Zur weiteren Betrachtung werden nun zwei Fälle unterschieden.

1. **Monotone Konvergenz:** Die Arbeit im Gebiet j ist nach dem Iterationsschritt größer als der Mittelwert der Arbeit über alle weiteren Gebiete ist.

$$W_j^{(k+1)} > \frac{1}{3} \sum_{i=1}^{3} W_i^{(k+1)} \tag{3.25}$$

2. **Alternierende Konvergenz:** Die Arbeit im j-ten Gebiet liegt unterhalb des Mittelwertes der anderen drei Gebiete. Dies hätte zur Folge, dass das Gebiet j nach dem Iterationsschritt nicht mehr das Gebiet mit

der größten Arbeit ist. Somit kehrt sich im zweiten Fall das Relationszeichen aus Ungleichung (3.25) um.

Da die Gesamtarbeit W genau der Summe aller Einzelarbeiten entspricht, lässt sie sich so formulieren:

$$\begin{aligned} & W && = && W_j^{(k)} + \sum_{i=1}^{3} W_i^{(k)} \\ \Leftrightarrow \quad & W_j^{(k)} && = && W - \sum_{i=1}^{3} W_i^{(k)} \end{aligned} \tag{3.26}$$

Im nächsten Schritt wird die rechte Seite der Ungleichung (3.24) betrachtet. Ersetzt man dort $W_j^{(k)}$ durch den Ausdruck in Gl. (3.26), so ergibt sich der folgende Wert:

$$\begin{aligned} W_j^{(k)} - \frac{1}{3}\sum_{i=1}^{3} W_i^{(k)} = & \quad W - \sum_{i=1}^{3} W_i^{(k)} - \frac{1}{3}\sum_{i=1}^{3} W_i^{(k)} \\ = & \quad W - \frac{4}{3}\sum_{i=1}^{3} W_i^{(k)} \end{aligned} \tag{3.27}$$

Diese Umformung gilt ebenfalls für den Iterationsschritt $k+1$, da sich an der Gesamtarbeit nichts ändert.

1. Monotone Konvergenz: Für Ungleichung (3.24) ergibt sich unter der ersten Bedingung und der Gl. (3.27) Ausdruck, da die Beträge vernachlässigt werden können:

$$\begin{aligned} & W - \frac{4}{3}\sum_{i=1}^{3} W_i^{(k+1)} && < && W - \frac{4}{3}\sum_{i=1}^{3} W_i^{(k)} \\ \Leftrightarrow \quad & \sum_{i=1}^{3} W_i^{(k+1)} && > && \sum_{i=1}^{3} W_i^{(k)} \end{aligned} \tag{3.28}$$

Wenn man nun die übertragene Arbeit $\delta W^{(k)}$ als die Arbeit betrachtet, die das Gebiet j an alle weiteren Gebiete abgibt, ergibt sich der folgende Aus-

druck:

$$\sum_{i=1}^{3} W_i^{(k+1)} = \sum_{i=1}^{3} W_i^{(k)} + \delta W^k \tag{3.29}$$

D.h. durch einsetzen von Gl. (3.29)in Ungleichung (3.28) erhält man, dass

$$\delta W^{(k)} > 0$$

ist. Unter der ersten Bedingung muss somit gelten, dass der Arbeitsübertrag von dem Gebiet mit der größten Arbeit in die anderen Gebiete positiv sein muss. Eine Beschränkung nach oben ist in diesem Fall nicht erforderlich.

2. Alternierende Konvergenz: Unter der zweiten Bedingung vertauschen sich beim Auflösen der Beträge auf der linken Seite der Ungleichung (3.24) die Vorzeichen. Der so entstandene Term kann mittels der bekannten Ergebnisse mit den folgenden Schritten umgeformt werden:

$$\begin{aligned}
& \frac{1}{3}\sum_{i=1}^{3} W_i^{(k+1)} - W_j^{(k+1)} && < \quad W_j^{(k)} - \frac{1}{3}\sum_{i=1}^{3} W_i^{(k)} \\
\overset{Gl.(3.27)}{\Leftrightarrow} \quad & \frac{4}{3}\sum_{i=1}^{3} W_i^{(k+1)} - W && < \quad W - \frac{4}{3}\sum_{i=1}^{3} W_i^{(k)} \\
\overset{Gl.(3.29)}{\Leftrightarrow} \quad & \frac{4}{3}\left(\sum_{i=1}^{3} W_i^{(k)} + \delta W^{(k)}\right) - W && < \quad W - \frac{4}{3}\sum_{i=1}^{3} W_i^{(k)} \\
\Leftrightarrow \quad & \delta W^{(k)} && < \quad \frac{3}{2}W - 2\sum_{i=1}^{3} W_i^{(k)}
\end{aligned}$$

Im Folgenden wird nun angenommen, dass die Gesamtarbeit W normiert ist, d.h. $W = 1$. Für die Abschätzung von $\delta W^{(k)}$ bedeutet das:

$$\Rightarrow \quad \delta W^{(k)} \quad < \quad \frac{3}{2} - 2\sum_{i=1}^{3} W_i^{(k)} \tag{3.30}$$

Die maximale zu übertragene Arbeit $\delta W^{(k)}$ ist somit von der Gesamtarbeit und von der Verteilung der Arbeit im Schritt k abhängig. Ist die Arbeit ausgeglichen, d.h. alle vier angrenzenden Gebiete besitzen gleich viel Arbeit, so erhält man als obere Schranke für $\delta W^{(k)}$ den Wert Null und es wird keine weiter Arbeit übertragen. Spätestens in diesem Fall wäre das Loadbalancing Verfahren abgeschlossen und müsste keine Arbeit mehr zu übertragen.

Probleme im Konvergenzverhalten können insbesondere bei lokalen Dichteunterschieden auftreten. Gibt es Stellen mit sehr hohen Dichte-Spitzen so ist es möglich, dass diese Spitzen durch eine zu groß gewählte Schrittweite nicht optimal erreicht werden. Das würde demnach heißen, dass auf einem sehr kleinen Gebiet Arbeit sehr konzentriert liegt. So kann es passieren, dass mehr Arbeit übertragen wird, als eigentlich beabsichtigt. Um solche Spitzen der Dichte abschätzen zu können wird die übertragene Arbeit $\delta W_i^{(k)}$ mittels der Dichte ausgedrückt:

$$\delta W_i^{(k)} = \int\limits_{\delta_x(i,i+1;\mu)} \rho(\vec{x}) d\vec{x} \tag{3.31}$$

Wenn man für die Betrachtung der Dichte zusätzlich noch voraussetzt, dass ausschließlich der mittlere Gitterpunkt im angegebenen System bewegt wird, so kann die veränderte Fläche, in Abbildung 3.21 farbig dargestellt, unter der Angabe der Eckpunkte berechnet werden.

Die Integralgrenze $\delta_x(i, i+1; \mu)$ beschreibt ein zweidimensionales Gebiet, welches genau die veränderte Fläche umfasst. In Abbildung 3.21 wird davon ausgegangen, dass die maximale Arbeit im Feld oben links liegt. Der Punkt $\vec{p}$ bezeichnet den linken und $\vec{q}$ den oberen Gitterpunkt. $\vec{x}$ ist der zu verschiebende Gitterpunkt in der Mitte des Systems. Wenn man davon ausgeht, dass der Gitterpunkt mit der Kraft $\vec{f}^{(k)}$ verschoben wird ergibt sich:

$$\vec{x}^{(k+1)} = \vec{x}^{(k)} + \mu \cdot \vec{f}^{(k)}$$

Die farbig eingezeichnete Fläche lässt sich in zwei Dreiecke aufteilen deren Eckpunkte jeweils bekannt sind. Der Flächeninhalt A lässt sich daher so

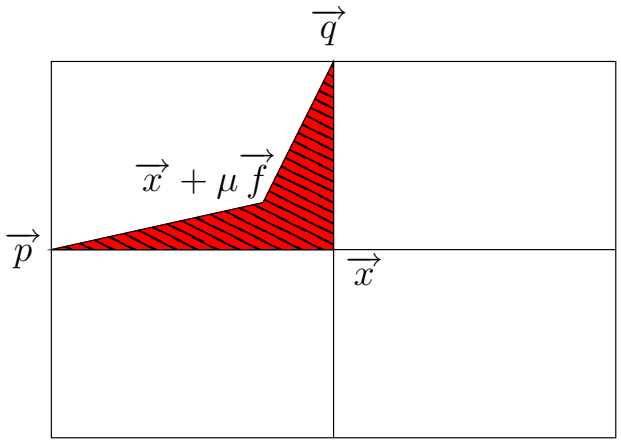


Abbildung 3.21: Überstrichene Fläche bei der Verschiebung

formulieren:

$$A\left(\vec{p}, \vec{q}, \vec{x}^{(i)}, \vec{x}^{(i+1)}\right) = \frac{1}{2}\begin{vmatrix} 1 & p_1 & p_2 \\ 1 & x_1^{(i)} & x_2^{(i)} \\ 1 & x_1^{(i+1)} & x_2^{(i+1)} \end{vmatrix} + \frac{1}{2}\begin{vmatrix} 1 & q_1 & q_2 \\ 1 & x_1^{(i)} & x_2^{(i)} \\ 1 & x_1^{(i+1)} & x_2^{(i+1)} \end{vmatrix} \tag{3.32}$$

Zur Vereinfachung gilt:

$$A\left(\mu, \vec{f}^{(i)}\right) \equiv F\left(\vec{p}, \vec{q}, \vec{x}^{(i)}, \vec{x}^{(i+1)}\right)$$

Wendet man nun den Mittelwertsatz der Integralrechnung an, so erhält man

$$\delta W_i^{(k)} := \int\limits_{\delta_x(i,i+1;\mu)} \rho(\vec{x}) d\vec{x} = \rho(\vec{\xi}) \cdot A\left(\mu, \vec{f}^{(i)}\right) \tag{3.33}$$

Hierbei liegt $\vec{\xi}$ innerhalb des verschobenen Gebietes, also innerhalb der gefärbten Fläche. Die Menge aller Punkte, die innerhalb dieser Fläche liegen, wird über den Ausdruck $\delta_x(i, i+1; \mu)$ definiert. Mit dem Ergebnis aus (3.33)

und der Ungleichung (3.30) ergibt sich folgende Abschätzung:

$$\begin{aligned} \rho(\vec{\xi}) \cdot A\left(\mu, \vec{f}^{(i)}\right) &< \frac{3}{2} - 2\sum_{i=1}^{3} W_i^{(k)} \\ \Rightarrow \max_{\vec{x} \in \delta_x(i,i+1;\mu)} \rho(\vec{x}) &< \frac{\frac{3}{2} - 2\sum_{i=1}^{3} W_i^{(k)}}{A\left(\mu, \vec{f}^{(i)}\right)} \end{aligned} \tag{3.34}$$

Die Ungleichung (3.34) gibt nun einen Zusammenhang zwischen der Dichte im betrachteten Gebiet und dem zu verwendenden Parameter μ an, der maßgeblich für die Verschiebung der Gitterpunkte verantwortlich ist. Man kann nun bestimmte Dichteverteilungen in Abhängigkeit der Variable μ zulassen oder μ so wählen, dass eine bestimmte Dichte mit dem Loadbalancing-Verfahren verarbeitet werden kann.

Für die weitere Betrachtung gilt ohne Beschränkung der Allgemeinheit, dass sich Abbildung 3.21 im Einheitsquadrat befindet. Zusätzlich gilt: $\vec{p} = (0\ ,\ 0.5)^T$ und $\vec{q} = (0.5\ ,\ 1)^T$. Darüberhinaus wirkt die Kraft $\vec{f}$ nach links oben. D.h. die 1. Komponente der Kraft $\vec{f}$ ist negativ und die 2. Komponente positiv. Um eine Abschätzung für den Parameter μ zu erhalten, wird Gl. (3.34) nach μ aufgelöst. Dazu ersetzt man den Ausdruck der Fläche $A\left(\mu, \vec{f}^{(i)}\right)$ durch die Definition aus Gl. (3.32). Wird Gl. (3.34) zuerst nach $A\left(\mu, \vec{f}^{(i)}\right)$ umgestellt

$$A\left(\mu, \vec{f}^{(i)}\right) < \frac{\frac{3}{2} - 2\sum_{i=1}^{3} W_i^{(k)}}{\max\limits_{\vec{x} \in \delta_x(i,i+1;\mu)} \rho(\vec{x})}$$

und im nächsten Schritt nach der Variablen μ aufgelöst, so ergibt sich die folgende obere Abschätzung für μ:

$$\mu < \frac{\frac{3}{2} - 2\sum_{i=1}^{3} W_i^{(k)}}{(-\frac{1}{4}f_1 + \frac{1}{2}x_1 f_2 + \frac{1}{4}f_2 - \frac{1}{2}x_1 x_2 f_2) \max\limits_{\vec{x} \in \delta_x(i,i+1;\mu)} \rho(\vec{x})} \tag{3.35}$$

Diese Abschätzung gilt unter der Bedingung, dass der Nenner der rechten Seite der Ungleichung (3.35) positiv ist. Da die Dichte positiv definit ist, muss noch gezeigt werden, dass der folgende Ausdruck für den betrachteten

Fall erfüllt ist:

$$-\frac{1}{4}f_1 + \frac{1}{2}x_1 f_2 + \frac{1}{4}f_2 - \frac{1}{2}x_1 x_2 f_2 > 0$$

Durch Umformung dieser Voraussetzung ergibt sich:

$$\begin{aligned} \Leftrightarrow \quad & \frac{1}{4}\left((2x_1 - 2x_1x_2 + 1)\cdot f_2 - f_1\right) && > && 0 \\ \Leftrightarrow \quad & \left(2x_1\left(1 - x_2\right) + 1\right)\cdot f_2 && > && f_1 \end{aligned} \tag{3.36}$$

Die Werte x_1 und x_2 beschreiben die Koordinaten des zu verschiebenden Gitterpunktes. Da dieser Gitterpunkt immer innerhalb des Einheitsquadrates liegt, gilt für seine Komponenten:

$$0 < x_1, x_2 < 1$$

Dies wiederum hat zur Folge, dass der Ausdruck $(2x_1\left(1 - x_2\right) + 1)$ in Gl. (3.36) stets positiv ist. Nach Voraussetzung ist f_2 zusätzlich positiv. Die linke Seite der Ungleichung (3.36) ist unter den gemachten Annahmen positiv. Die rechte Seite beinhaltet nur die Komponente f_1 und ist nach Voraussetzung negativ. Folglich ist die Ungleichung für die gegebenen Vorrausetzungen erfüllt. Letztendlich folgt daraus, dass die Abschätzung aus der Ungleichung (3.35) für den Parameter μ verwendet werden kann.
Wenn der Schwerpunkt in einem anderen der drei Gebiete liegt, so wäre die Vorgehensweise analog. Unter gleichen Voraussetzungen könnte eine obere Schranke für μ bestimmt werden, da diese Fälle symmetrisch zu dem betrachteten Fall sind.

3.5.3 Einhalten einer gegebenen Fehlerschranke

Nach dem im Kapitel 3.5.1 gezeigt wurde, dass mithilfe des Matrixoperators die Arbeit ausgeglichen werden kann, beschäftigt sich dieses Kapitel mit der Frage, wie lange man iterieren muss um mit Sicherheit eine gegebene Fehlerschranke einzuhalten. D.h. wie viele Iterationsschritte müssen für diesen Zweck durchgeführt werden bzw. wie oft muss man die Matrix $A_\omega\left(\gamma\right)$

für $\gamma \geq 56$ auf den Arbeitsvektor anwenden, um einen Arbeitsunterschied kleiner als die vorgegebene Schranke zu erreichen. Dazu wird die Gl. (3.22) betrachtet unter der Voraussetzung, dass die Werte für α_i normiert sind und somit gilt:

$$\alpha_1 = 1,\ \alpha_i < 1,\ \forall i \in \{2, \ldots, n\} \tag{3.37}$$

Bekannt ist, dass der Arbeitsvektor nach l Iterationsschritten wie folgt geschrieben werden kann:

$$\vec{w}^{(l)} = \sum_{i=1,..,n} \left(\alpha_i \cdot \lambda_i^l \cdot \vec{e}_i\right)$$

Wenn man zudem davon ausgeht, dass λ_1 der größte Eigenwert ist, also der einzige Eigenwert mit dem Betrag 1 und $\vec{e}_1$ der zugehörige Eigenvektor ist, so kann man den dazugehörigen Summanden abspalten und für den restlichen Teil die angegebene Abschätzung verwenden:

$$\vec{w}^{(l)} = \alpha_1 \cdot \vec{e}_1 + \sum_{i=2,..,n} \left(\alpha_i \cdot \lambda_i^l \cdot \vec{e}_i\right)$$

Nach (3.37) und unter der Annahme das λ_2 der zweitgrößte Eigenwert ist, gilt nun die folgende Abschätzung für die restlichen Summanden:

$$\left|\sum_{i=2,..,n} \left(\alpha_i \cdot \lambda_i^l \cdot \vec{e}_i\right)\right| \leq \left|\sum_{i=2,..,n} \lambda_i^l \cdot \vec{e}_i\right| < \left|(n-1) \cdot \lambda_2^l\right| \tag{3.38}$$

Der gemachte Fehler im l-ten Iterationschritt kann nun nach oben durch $(n-1) \cdot \lambda_2^l$ abgeschätzt werden. Wenn man die Fehlerschranke mit 10^{-p} vorgibt, so ergibt sich für den Fehler die folgende obere Abschätzung:

$$\begin{aligned} (n-1) \cdot \lambda_2^l &\geq 10^{-p} \\ \Leftrightarrow \qquad l &\geq \frac{-p - \log_{10}(n-1)}{\log_{10}(\lambda_2)} \end{aligned} \tag{3.39}$$

Diese *a priori* Abschätzung stellt sicher, dass die Fehlerschranke immer ein-

gehalten wird. Wie scharf diese Abschätzung ist, wurde bis hier jedoch noch nicht betrachtet. Die Abbildung 3.22 zeigt die Fehlerentwicklung für Systeme unterschiedlicher Größen. Der Fehler soll hier immer kleiner als 10^{-4} sein. Die blau gestrichelte Linie zeigt die Fehlerschranke, welche nicht überschritten werden darf. Die x-Achse gibt Auskunft darüber, in wieviele Gebiete das System in jeder Dimension unterteilt ist. Beispielsweise ist für $x = 8$ das System insgesamt in $8 \cdot 8 \cdot 8 = 512$ Gebiete unterteilt. Die grüne Kurve bzw. die roten Punkte geben an, welcher Fehler tatsächlich gemacht wird, wenn man so viele Iterationsschritten macht, so dass die Ungleichung (3.39) erfüllt ist.

Die Abbildung 3.22 verdeutlicht, dass die Abschätzung zwar die gegebene

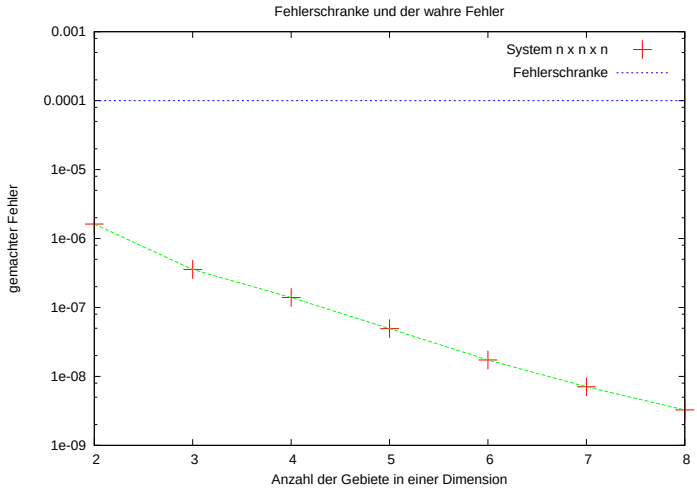


Abbildung 3.22: Obere Schranke und der wirklich gemachte Fehler

Schranke einhält, aber insbesondere für große Systeme nicht sehr scharf ist. Dies ist dadurch zu erklären, dass in Gl. (3.38) alle Eigenwerte kleiner als eins durch den zweitgrößten Eigenwert λ_2 abgeschätzt werden. Eine scharfe Abschätzung wäre somit nur dann gegeben, falls für alle Eigenwerte außer λ_1 gilt $\lambda_2 = \ldots = \lambda_n$. In allen weiteren Fällen wird die Abschätzung ungenauer je größer die Abweichung aller weiteren Eigenwerte vom zweiten größten Eigenwert λ_2 ist.

Kapitel 4

Resultate

In diesem Kapitel werden die Ergebnisse von den in Kapitel 3 beschriebenen Verfahren und Vorgehensweisen gezeigt und gegeneinander abgegrenzt.

Im ersten Teil wird der zweidimensionale Fall mit der nicht-hierarchischen Gebietsaufteilung betrachtet. Neben Ergebnissen zum Arbeitsausgleich der Gebiete werden zusätzlich die Laufzeiten der zwei Verfahren zum Test auf Gebietszugehörigkeit, „Vektorbasierter Test“ und die „Rücktransformation der Reference Shape Transformation“, verglichen. Im letzten Teil wird ein Ergebnis für ein dreidimensionales System angegeben und das hierarchische Modell im Vergleich zum nicht-hierarchischen Modell betrachtet.
Für die Betrachtung der Systeme kann man zwischen aktiver und passiver Teilchenbewegung unterscheiden. Die aktive Teilchenbewegung beschreibt das physikalische Bewegen von Teilchen. Ein Teilchen, welches physikalisch nicht bewegt wird, sondern nur durch die Verschiebung der Gebietsgrenzen seine Gebietszugehörigkeit ändert, vollzieht eine passive Bewegung. Die hier betrachteten Resultate beziehen sich ausschließlich auf ruhende Systeme, d.h. dass Teilchen keine aktive Bewegungen ausführen.

4.1 Zweidimensionale Lastbalance

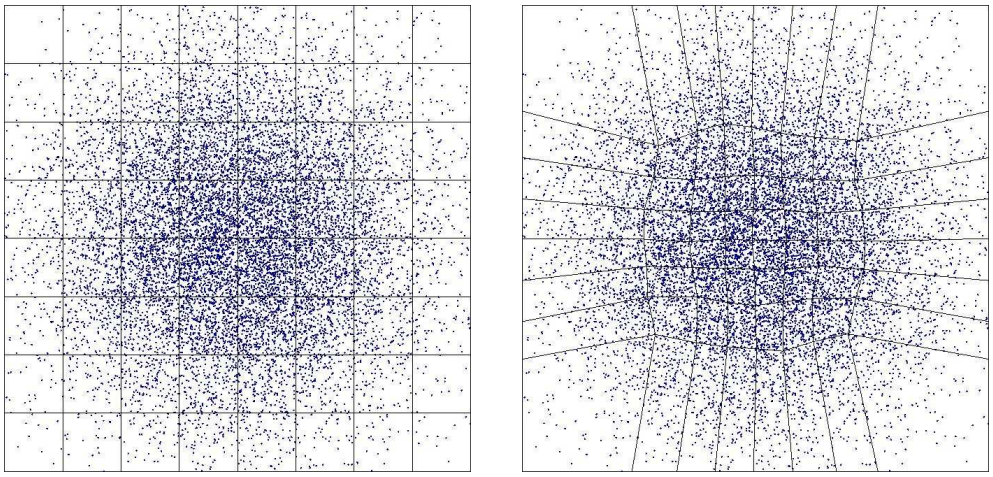
Zur Bestimmung des Konvergenzverhaltens wurden verschiedene Tests durchgeführt. Für diese Tests wurden physikalische Teilchen in ein System gelegt. Bei den beschriebenen Test wird davon ausgegangen, dass diese Teilchen sich nicht bewegen. Ausschließlich die Gitterpunkte werden bewegt, um zu sehen, wie sich die Verteilung der Teilchen auf die einzelnen Gebieten ändert und ausgleicht. Als Maß für den Ausgleich zwischen den einzelnen Gebieten wird die empirische Varianz betrachtet.

$$Var = \frac{1}{g} \cdot \sum_{p=1}^{g} (n_p - \overline{n})^2 \tag{4.1}$$

wobei gilt:
g: Anzahl aller Gebiete bzw. Prozessoren im System
n_p: Anzahl der Teilchen im Gebiet p
$\overline{n}$: Mittelwert der Anzahl der Teilchen über alle Gebiete
Bei den betrachteten Gebieten werden keine aktiven Bewegungen der physikalischen Teilchen berücksichtigt. Daher werden auch keine Interaktionen zwischen den Teilchen berechnet. Als Maß für die Arbeit wird aus diesem Grund nur die Anzahl der Teilchen in den Gebieten betrachtet.

Normalverteilte Teilchen

In diesem Kapitel wird davon ausgegangen, dass die Teilchen normalverteilt in dem betrachteten System liegen. Für zwei Dimensionen zeigt die Abbildung 4.1 die Lage der Teilchen und der Gittergrenzen zu Beginn der Simulation und nach dem Loadbalancing Schritt. In dem System der Länge $L \times L$ liegen die physikalischen Teilchen normalverteilt um den Punkt $\left(\frac{L}{2}, \frac{L}{2}\right)$. Zu Beginn wird ein äquidistantes Gitter über das System gelegt. Das hier betrachtete Gitter ist ein Gitter der Form 8×8. Die Abbildung 4.1 zeigt, dass sich die Gebiete, an den Stellen zusammen ziehen, wo sich besonders viele Teilchen befinden.
Zur mathematischen Betrachtung wird nun die bereits beschriebene Varianz



(a) Gitteraufteilung zu Beginn (b) Ausbalancierte Gitterpunkte

Abbildung 4.1: Gebietszerlegung zu Beginn und nach der Verschiebung

der Anzahl der Teilchen der einzelnen Gebiete betrachtet. Diese Varianz, welche in der Gl. (4.1) für das in dieser Arbeit betrachtete Problem definiert wurde, wird in der Abbildung 4.2 logarithmisch über den Iterationsschritten aufgetragen. Dabei wurden insgesamt vier Systeme betrachtet, für die jeweils die Varianz in jedem Iterationsschritt bestimmt wurde. Der rote Graph zeigt die Varianz für ein System mit 25.600 Teilchen. Dies würde einer mittleren Anzahl von 400 Teilchen pro Gebiet entsprechen. Zu Beginn startet der Loadbalancing-Schritt mit einer Varianz von über 160.000. Diese Varianz fällt bis zum Iterationsschritt 200 exponentiell ab bis auf einen Wert von 6. Ein ähnlicher Verlauf der Varianzen ist ebenfalls für die Systeme mit 38.400, 64.000 und 96.000 Teilchen zu erkennen. Je mehr Teilchen das betrachtete System beinhaltet, desto schneller fällt die Varianz des Systems ab. Diese Varianz bleibt von diesem Zeitpunkt an fast unverändert. Das System mit 64.000 Teilchen, welches bei einer Varianz von über 1 Millionen startet, konvergiert in der Varianz gegen den Wert eins. Das System mit 96.000 Teilchen, dies entspricht durchschnittlich 1500 Teilchen pro Prozessor, startet bei einer Varianz von über 2.3×10^6. Der exponentielle Abfall der Varianz ist hier so stark, dass bereits im 70. Iterationsschritt eine Varianz von etwa zehn erreicht wird.

Im optimalen Fall kann die Varianz den Wert Null annehmen, was bedeutet, dass der Ausgleich der Teilchen perfekt funktioniert. Dieser optimale Austausch der Arbeit zwischen den Gebieten ist in der Regel jedoch nicht gegeben, da die Stabilitätskriterien und die daraus resultierenden Einschränkungen nicht jede beliebige Form der Gebiete zulassen, wie es eventuell für den bestmöglichen Arbeitsausgleich notwendig wäre.
Das hier betrachtete Beispiel ist jedoch kein Fall der in der Praxis auftritt. Dies liegt an den beiden folgenden Punkten:

- Wenn man Teilchensysteme oder Polymere rechnet, so sind die Teilchen nie normalverteilt, wie in diesem Beispiel angenommen. Die Normalverteilung der Teilchen wurde hier angenommen, um an einem anschaulichen Beispiel die Idee des gradienten basierten Lastbalance-Verfahren zu zeigen und den Umgang mit der Varianz als Maß für den Arbeitsausgleich zu demonstrieren.

- In der praktischen Anwendung würde man keineswegs die Iterationen immer bis zum 300. Schritt durchlaufen lassen. In der Anwendung ist man zwar daran interessiert, dass die Arbeit des Systems ausgeglichen wird, es ist jedoch nicht unbedingt notwendig, das absolute Minimum zu erreichen. Wenn man bei einem System mit 96.000 Teilchen und 64 Prozessoren eine Varianz von 10 erhält, so wäre dies bereits eine gute Approximation für eine gleiche Arbeitsverteilung und das Iterationsverfahren könnte abgebrochen werden. Der folgende Abschnitt beschäftigt sich nun mit der Frage, unter welchen Umständen welche Anzahl an Iterationsschritten sinnvoll ist, um eine sinnvolle Arbeitsaufteilung zu erreichen.

Abbruchkriterium für das Loadbalancing-Verfahren

Die Abbildung 4.2 zeigt ein typisches Verhalten des hier behandelten Loadbalancing -Verfahrens. Während der ersten Schritte des Verfahrens fällt die Varianz und somit der Arbeitsunterschied zwischen den Gebieten sehr rasch

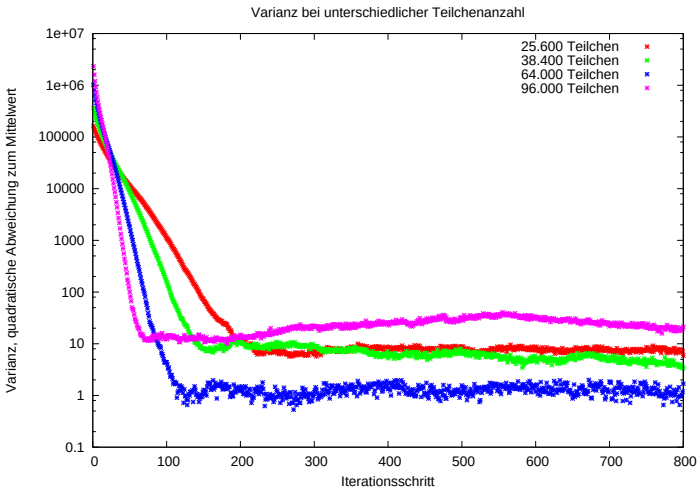


Abbildung 4.2: Varianzen bei unterschiedlicher Teilchenanzahl

ab. Nach diesem steilen Abfall, nimmt die Varianz nur noch wenig oder gar nicht mehr ab. Daher ist es sinnvoll, das Verfahren abzubrechen sobald man, anschaulich gesprochen, von dem Teil des steilen Abfalls in den Teil des weniger steilen Abfalls übergeht. Zur Erkennung dieses Übergangs betrachtet man den Quotienten aus Gl. (4.2).

$$\frac{V^{(i+1)}}{V^{(i)}} \tag{4.2}$$

In dieser Formel beschreibt $V^{(i+1)}$ den Wert der Varianz im $(i+1)$-ten Iterationsschritt und $V^{(i)}$ die Varianz im Iterationsschritt i. Unter der Annahme, dass die Varianz mit steigender Anzahl an Iterationsschritten abfällt, nimmt der Quotient einen Wert kleiner eins an. Ein sinnvolles Abbruchkriterium ergibt die folgende Ungleichung:

$$\frac{V^{(i+1)}}{V^{(i)}} \leq 1 - \alpha$$

Verringert sich die Varianz vom i-ten Iterationsschritt zum $(i+1)$-ten Iterationsschritt um weniger als den Anteil α, so wird die Iteration bei diesem

Zeitschritt abgebrochen. Die Iteration wird jedoch weiter fortgeführt, falls die Änderung der Varianz größer ist als α.
Das hier beschriebene Abbruchkriterium impliziert, dass die Varianz in jedem Zeitschritt abfällt. Man könnte sich jedoch vorstellen, dass für einzelne Zeitschritte die Varianz nicht abfällt, obwohl das Verfahren noch nicht auskonvergiert ist. Um das Verfahren in diesem Fall nicht direkt fälschlicherweise abzubrechen, ist es auch möglich den Quotienten aus Gl. (4.2) über mehrere Zeitschritte zu betrachten. D.h. statt $V^{(i+1)}$ könnte man das laufende Mittel der Varianz betrachten, d.h. den Mittelwert der n „Vorgänger-Varianzen“ benutzen. Die so entstandenen $V^{*(i)}$ und $V^{*(i+1)}$ lassen sich dann folgendermaßen definieren:

$$V^{*(i)} = \frac{1}{n} \sum_{i=1}^{n} V^{(i-n)}$$

$$V^{*(i+1)} = \frac{1}{n} \sum_{i=1}^{n} V^{(i+1-n)}$$

4.2 Vergleich: Vektorbasierter Test und „Reference Shape Transformation“

Im Kapitel 3.1.1 wurde die Rücktransformation der „Reference Shape Transformation“ vorgestellt, um Teilchen auf ihre Gebietszugehörigkeit zu testen. Dieses Verfahren wurde jedoch nur für den zweidimensionalen Fall verwendet. Zusätzlich wurde in Kapitel 3.1.2 der vektorbasierte Test erarbeitet. Dieser Test wird in dieser Arbeit für zweidimensionale und für dreidimensionale Systeme verwendet.
Da beide Methoden für den zweidimensionalen Fall verwendet werden können, werden in diesem Kapitel die Laufzeiten verglichen. Für diesen Vergleich wird der „Laufzeitquotient“ (LQ)betrachtet:

$$LQ = \frac{t_{vs}}{t_{rs}}$$

Die Variable t_{rs} beschreibt die Zeit, die benötigt wird um mit der „Reference Shape Transformation“ die Gebietszugehörigkeit zu testen. Für das gleiche System und der gleichen Anzahl an Prozessoren beschriebt t_{vs} die Zeit, die durch den „vektorbasierten Test“ verbraucht wird. In der Abbildung 4.3 wird dieses Verhältnis dargestellt. Die Werte auf der x-Achse geben an, wie viele Teilchen im Mittel in jedem Gebiet liegen. Für jeden dieser Werte wird auf der y-Achse das Verhältnis LQ angegeben. Die Graphik zeigt, dass der

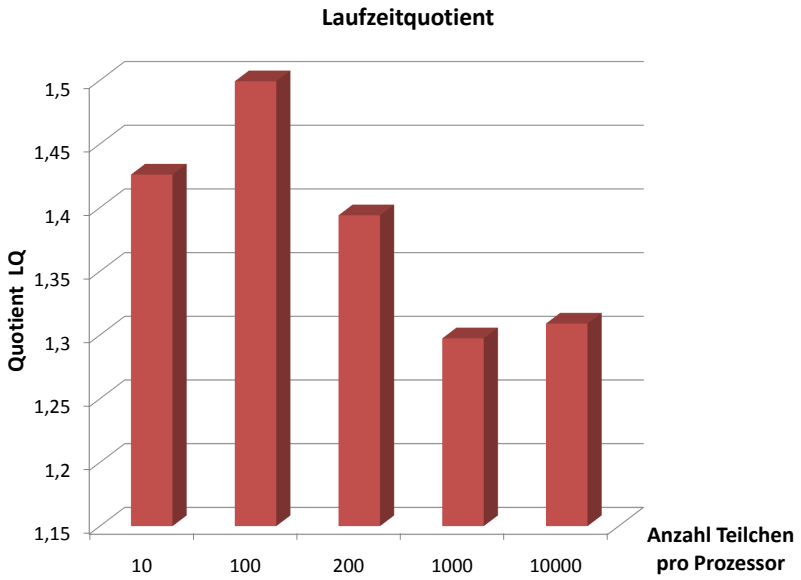


Abbildung 4.3: Der Quotient LQ für unterschiedliche Teilchenanzahlen pro Prozessor

Quotient für die betrachteten Systeme größer eins ist. Daraus folgt, dass die Rücktransformation der „Reference Shape Transformation“ das Problem schneller verarbeitet, als der „vektorbasierte Test“. Für ein System bei einer Gebietsaufteilung, bei der im Mittel zehn Teilchen in einem Gebiet liegen, liegt der Quotient bei ca. $1,4$. Dieser Wert für LQ verändert sich für größere Teilchenanzahlen pro Gebiet nur wenig. Für 100 Teilchen pro Prozessor liegt er bei ca. $1,5$. Wenn man davon ausgeht, dass durchschnittlich 1000 oder 10000 Teilchen auf einem Gebiet liegen, so beträgt LQ ca. $1,3$. Für die hier getesteten Beispiele benötigt der „vektorbasierte Test“ ungefähr 30% bis 50

% mehr Zeit als die Vorgehensweise mittels „Reference Shape Transformation“. Dieses Ergebnis gilt jedoch ausschließlich für den zweidimensionalen Fall. Da die Rücktransformation der „Reference Shape Transformation“, wie in Kapitel 3.1.1 beschrieben, für drei Dimensionen nicht weiter verwendet wird, werden an dieser Stelle keine Ergebnisse für den dreidimensionalen Fall angegeben.

4.3 Dreidimensionale Lastbalance

Die Ergebnisse für drei Dimensionen sind im Wesentlichen ähnlich zu denen aus dem zweidimensionalen Fall. Die Abbildungen 4.4 und 4.5 zeigen ein System mit 25.000 Teilchen. Dieses System ist im Gegensatz zu den Systemen aus dem vorrangegangenen Kapitel ein Beispiel aus der praktischen Anwendung. In der Abbildung 4.4 ist die äquidistante Unterteilung der Gebiete zu Beginn des Loadbalancings zu erkennen. Das System ist in jede Richtung in acht Gebiete unterteilt bei eine Ausdehnung von $75 \times 75 \times 450$. Der

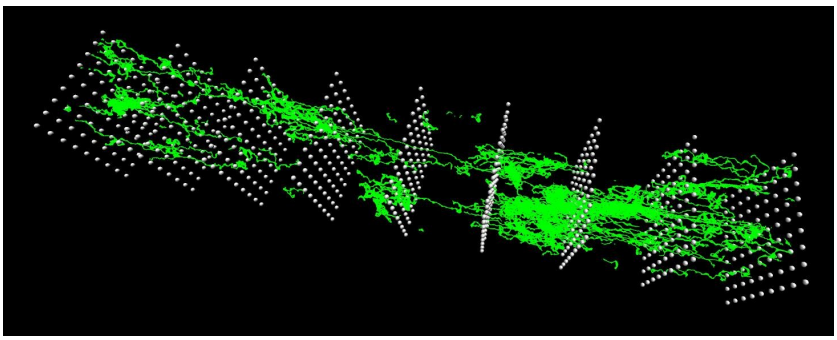


Abbildung 4.4: System vor dem Loadbalancing

Loadbalancing-Schritt des hier betrachteten Systems von Polymeren wurde ohne Berücksichtigung des hierarchischen Modells durchgeführt. Der Verlauf der Varianz bzgl. der Iterationsschritte ist ähnlich zu dem Verlauf im zweidimensionalen. In Abbildung 4.6 ist jedoch zu erkennen, dass die Varianz zu Beginn erst ansteigt, bevor sie dann abfällt. Da man mit Massenschwerpunkten arbeitet, hat man keine genaue Vorhersage wieviel Arbeit in jedem

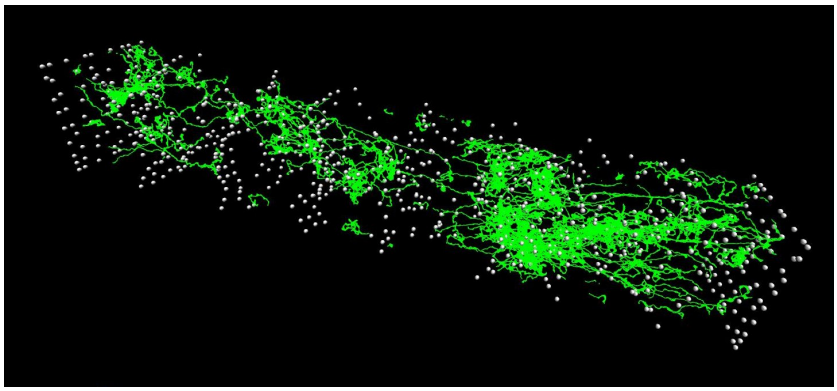


Abbildung 4.5: System nach dem Loadbalancing

Iterationsschritt verschoben wird. Somit kann es passieren, dass der Arbeitsausgleich nicht während aller Iterationsschritten eindeutig zu einer Abnahme der Varianz führt.

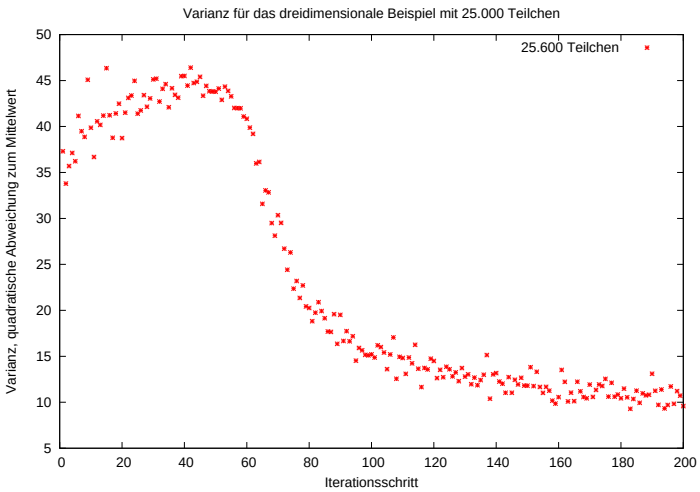


Abbildung 4.6: Varianz für das betrachtete dreidimensionalen System

4.4 Das hierarchische Modell

Im Kapitel 4.1 wurde in Gl. (4.1) die Varianz als Maß für den Arbeitsausgleich definiert. Wenn man die Betrachtung für den Arbeitsaugleich für eine feste Anzahl an Prozessoren definiert, so macht diese Definition Sinn. Erhöht man jedoch die Anzahl der Prozessoren, so wird die Varianz für eine feste Teilchenzahl unweigerlich kleiner. Diese Verringerung der Varianz bei Erhöhung von Prozessorzahlen wird somit nicht zwangsläufig durch einen besseren Arbeitsausgleich hervorgerufen. Um den Arbeitsausgleich für gleichbleibende Teilchenanzahlen bei unterschiedliche Anzahlen an Prozessoren besser vergleichen zu können, wird das neue „Ausgleichsmaß“ AG wie folgt definiert:

$$AG = \frac{1}{g} \cdot \sum_{p=1}^{g} \left(\frac{n_p}{\overline{n}} - 1 \right)^2 \tag{4.3}$$

Durch diese Änderung des Maßes für den Arbeitsausgleich erreicht man, dass das „Ausgleichsmaß“ für große Prozessorzahlen nicht abnimmt, ohne dass ein Arbeitsausgleich stattfindet. Im weiteren Verlauf dieses Kapitels wird mit Gl. (4.3) als Maß für den Arbeitsausgleich gearbeitet. Die hier eingeführte Definition macht insbesondere Sinn für das hierarchische Modell, da dort die Anzahl der Prozessoren zunimmt, die Teilchenanzahl aber konstant bleibt. Zunächst wird an einem allgemeinen Beispiel das hierarchische Modell motiviert. Zu diesem Zweck zeigt die Abbildung 4.7 den Verlauf des Arbeitsausgleiches für ein System mit 7.000 Teilchen unter der Verwendung von 16 Prozessoren, 64 Prozessoren und 256 Prozessoren ohne Einsatz des hierarchischen Modells. Es ist zu erkennen, dass der Arbeitsunterschied für eine kleinere Anzahl an Gebieten schneller abfällt.

Da der Arbeitsunterschied für 256 Prozessoren nur sehr langsam abnimmt, wird nun zusätzlich das hierarchische Modell betrachtet. Die Abbildung 4.8 zeigt den Verlauf für den Arbeitsausgleich mit und ohne Verwendung des hierarchischen Modells. Insgesamt werden 4000 Iterationen für beide Fälle durchgeführt. Das hierarchische Modell startet mit einer Unterteilung in vier Gebiete und wird über 16 und 64 Gebiete bis auf 256 Gebiete erhöht. Das

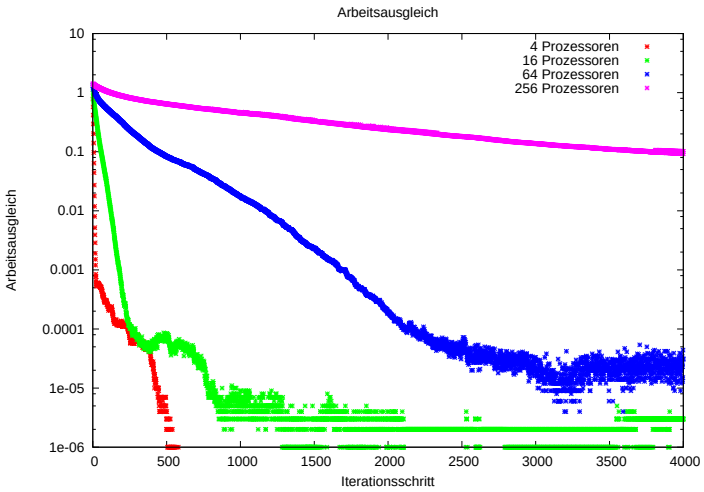


Abbildung 4.7: Varianzen für unterschiedliche Prozessoranzahlen

nicht hierarchische Modell startet von Beginn an mit 256 Gebieten. Deutlich zu erkennen ist, dass der Arbeitsausgleich im nichthierarchischen Fall deutlich langsamer ist als in fast jedem Schritt des hierarchischen Modells. In jedem Schritt des hierarchischen Modells werden die Arbeitsunterschiede vergleichsweise schnell ausgeglichen. Nach 4000 Iterationen hat das hierarchische Modell ein „Ausgleichsmaß“ von fast 0.001 für 256 Gebiete erreicht. Während des gleichen Iterationsschritts liegt das „Ausgleichsmaß“ für das nicht hierarchische Modell noch bei ca. 0.1. Dieser deutliche Unterschied zeigt, dass das Loadbalancing-Verfahren mit Anwendung des hierarchischen Modells für stark inhomogene Systeme wesentlich effektiver ist.

Im folgenden Beispiel wird das Loadbalancing-Verfahren auf ein weiteres System angewendet. Anhand dieses Beispiels wird der Unterschied zwischen dem hierarchischen und nicht-hierarchischen Modell nocheinmal anschaulich erläutert. Für dieses System liegen die physikalischen Teilchen fast ausschließlich im oberen linken Viertel des Gebietes. Im ersten Schritt des hierarchischen Modells startet man mit vier Gebieten. Diese Gebiete werden so verändert, dass die Arbeit in den unterschiedlichen Gebieten fast gleich ist. Dies

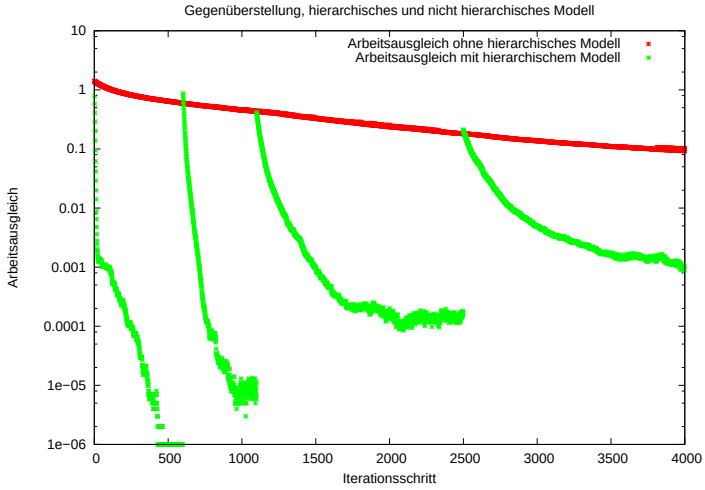
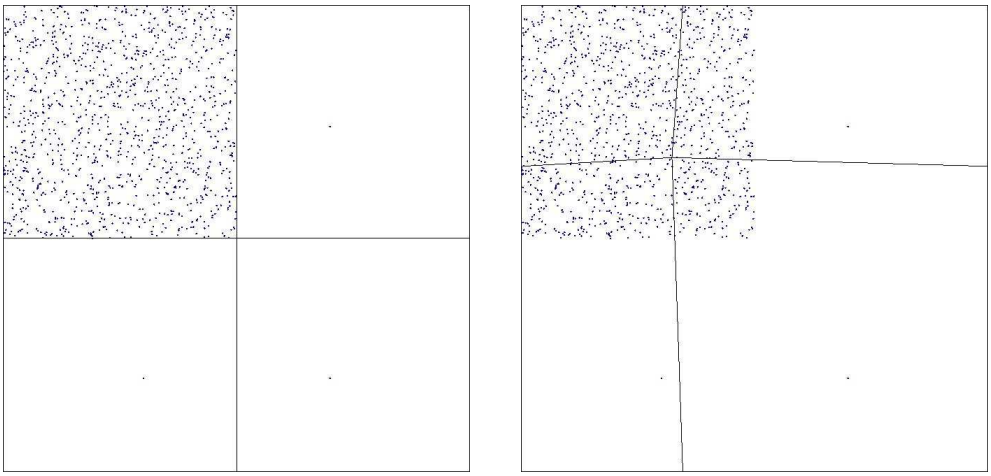


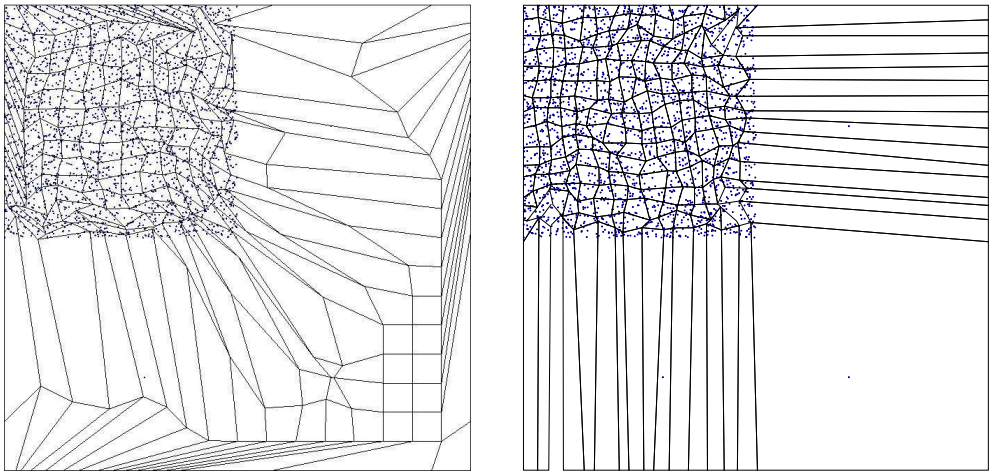
Abbildung 4.8: Varianzen im Vergleich



(a) Gitteraufteilung zu Beginn

(b) Gebietsaufteilung nach dem ersten Hierarchieschritt

Abbildung 4.9: Gebietszerlegung im 1. Hierarchieschritt



(a) Gebietszerlegung ohne das hierarchische Modell

(b) Gebietszerlegung mit hierarchischem Modell

Abbildung 4.10: Gebietszerlegung nach der gleichen Anzahl von Iterationen

hat zur Folge, dass die beweglichen Gitterpunkte in den Teil des Systems verschoben werden, in dem sich fast alle Teilchen befinden, wie in Abbildung 4.9 zu sehen ist. Im nächsten Schritt wird jedes der vier Gebiete in vier weitere Gebiete unterteilt. Die so entstandenen 16 Untergebiete werden nun abermals bezüglich ihrer Teilchenanzahl nach dem vorgestellten Verfahren ausgeglichen. Diese Verfeinerung der Gebiete kann so oft weitergeführt werden, bis die gewünschte Anzahl an Gebieten und somit an Prozessoren erreicht ist. Auch in diesem Beispiel werden vier Hierarchieschritte durchlaufen, so dass das Gebiet in der letzten Stufe in 256 Gebiete unterteilt ist. Für das gleiche Teilchensystem wurde zusätzlich der nicht hierarchische Fall betrachtet. Diese Simulation wurde bereits zu Beginn mit 256 Gebieten gestartet. Die Abbildung 4.10 zeigt das Ergebnis für das hierarchische Modell und für das nicht-hierarchische Modell nach jeweils 7000 Iterationsschritten. Die Abbildung 4.10(a) zeigt die Gebietsaufteilung für das nicht-hierarchische Modell. Dort ist zu erkennen, dass ca. 70 Gebiete noch keine Teilchen beinhalten und das System noch nicht auskonvergiert ist. Die Gebietsaufteilung des hierarchischen Modells in Abbildung 4.10(b) zeigt hingegen eine sehr gute Aufteilung der Teilchen auf die verschiedenen Gebiete.

D.h., wenn von Beginn an alle Prozessoren auf einem äquidistanten Gitter arbeiten, kann die Lastbalance wesentlich schlechter sein als beim hierarchischen Modell. Dies ist insbesondere dann der Fall, wenn Teilchen stark inhomogen verteilt sind. Hierbei gibt es zu Beginn der Simulation viele Prozessorgebiete, die keine Teilchen besitzen. Wenn zusätzlich auch ihre Nachbargebiete keine Teilchen beinhalten, so kann es unter Umständen sehr lange dauern, bis die Eckpunkte des ersten leeren Gebietes so verändert werden, dass das ursprünglich teilchenlose Gebiet so verschoben wird, dass es Teilchen beinhaltet. Die daraus resultierende sehr langsame Konvergenz kann dann mittels des hierarchischen Modells wesentlich verbessert werden. In Abbildung 4.10(a) ist zusätzlich noch zu erkennen, dass einige Gebiete im unteren rechten Teil des Systems noch äquidistant sind. Daran wird deutlich, dass diese Gebiete bis zum betrachteten Iterationsschritt noch gar nicht verändert worden sind, da keines ihrer Nachbargebiete Punkte beinhaltet und somit die Gittergrenzen noch nicht verschoben wurden. Die auffälligen Verzerrungen der leeren Gebiete am Rand, sind auf die Randbedingungen zurückzuführen. Sie besagen, dass beispielsweise die Verschiebung der Gitterpunkte auf dem unteren Rand von der Teilchenanzahl der obersten Gebiete abhängig sind, was dazu führt, dass die Verschiebung der Gitterpunkte auf der oberen Kante identisch zu den Verschiebungen auf der unteren Kante ist (periodische Randbedingung). So kommt es dazu, dass die Gitterpunkte des unteren Randes alle in die linke Hälfte gezogen werden, da in der linken Hälfte der obersten Gebietsreihe einige der Teilchen liegen, während die ersten inneren Punkte oberhalb des unteren Randes weiterhin in der unteren linken Hälfte verharren. Analog verhält es sich mit den Verzerrungen am rechten Rand des Systems. Abschließend bleibt zu sagen, dass durch den hierarchischen Ansatz die Transportgeschwindigkeit der Gitterpunkte in die Region mit höherer Teilchendichte wesentlich erhöht wird.
Die in diesem Kapitel für den zweidimensionalen Fall betrachteten Verteilungen der Teilchen beschreiben keine in der Natur vorkommenden Gegebenheiten. Sie fungieren als anschauliche Beispiele für die Arbeitsweise des „Gradientenbasierten Lastbalance-Verfahren“.

Kapitel 5

Zusammenfassung und Ausblick

Dieses zusammenfassenden Kapitel enthält eine Bewertung des Verfahrens mithilfe der Daten, welche im Kapitel 4 untersucht wurden. Darüber hinaus werden Erweiterungen und eventuelle Verbesserungen vorgestellt. Zudem werden weitere Punkte angesprochen, die bei einer umfassenden Bewertung in Zukunft noch berücksichtigt werden müssen.

5.1 Bewertung

In dieser Arbeit wurden nur Systeme betrachtet, bei denen sich die Teilchen des physikalischen Systems nicht bewegen. Für solche „ruhenden Systeme“ ist der Ausgleich der Arbeit mit dem Verfahren dieser Arbeit möglich. Selbst große Unterschiede können ausgeglichen werden. Betrachtet man dazu beispielsweise die Abbildung 4.2, so wird deutlich, dass die Varianz und somit das Ungleichgewicht der Arbeiten auf den einzelnen Gebieten z.T. von 10^6 auf weniger als 10 verringert werden kann. Eine wichtige Rolle dabei spielte die Frage, wie es möglich ist, diese Verringerung der Varianz bzw. des „Ausgleichsmaßes“ *AG* zu beschleunigen. Der dafür verwendete „Hierarchische Ansatz“ zeigte, dass er für sehr inhomogene Verteilungen eine gute Optimierung der Verringerung der Iterationsanzahl darstellt. Insgesamt ist festzustellen, dass die Ergebnisse die gewünschten Ziele erreichen. Ein weiterer Vorteil (ins-

besondere im Vergleich zum Verfahren von Halver siehe Kapitel 2.5.2) besteht darin, dass die Nachbarschaftsbeziehungen immer erhalten bleiben und somit keine Prüfungen auf wechselnde nächste Prozessoren nötig sind. Dies beinhaltet zusätzlich, dass auch die Anzahl der Nachbarprozessoren konstant 3^d ist. Wichtig ist außerdem die Wahl der Methode, die entscheidet, ob ein Teilchen außerhalb oder innerhalb eines Gebietes liegt. Auch hier wurden zwei Ansätze, die Reference Shape Transformation in Kapitel 3.1.1 und die Berechnung mittels Vektoren in Kapitel 3.1.2, aufgezeigt. Das folgende Kapitel zeigt weitere Fragestellungen, die über diese Arbeit hinaus gehen und erläutert Probleme, die in dieser Arbeit noch nicht oder noch nicht tiefgreifend genug untersucht wurden.

5.2 Ergänzungen und Erweiterungen

In diesem Kapitel werden Ideen für mögliche Änderungen oder Erweiterungen gegeben. Man könnte sich beispielsweise vorstellen, den Test auf Gebietszugehörigkeit anders anzugehen und weiter Methoden entwickeln, die erkennen, ob ein Teilchen innerhalb oder außerhalb eines Gebietes liegt.
Man könnte außerdem das in dieser Arbeit verwendete hierarchische Modell, welches für eine schnellere Konvergenz sorgt anders formulieren, z.B. mit einer anderen Aufteilung der Gebiete im nächsten Hierarchieschritt. Außerdem wäre es möglich die Stabilitätseigenschaft und somit die Constraints anders zu formulieren.
Keine Beachtung findet in dieser Arbeit die Kommunikation zwischen den einzelnen Prozessoren. Sie ist aufgrund der Nachbarschaftseigenschaften zwar nicht sehr aufwändig müsste aber für eine umfassende Bewertung zusätzlich berücksichtigt werden. Weiterhin müssen die hier erarbeiteten Grundlagen noch in größere Programmpakete, insbesondere in das Paket MP2C [14], eingebaut werden.
In der Anwendung wird zudem die Interaktion zwischen den eigentlich physikalischen Teilchen berechnet. Das heißt aber auch, dass sogenannte „Geisterteilchen“ berücksichtigt werden müssen. Die Abbildung 5.1 zeigt eine zwei-

dimensionale Aufteilung in vier Gebiete. Die schwarzen Linien bilden dabei die Gebietsgrenzen. Die in rot eingezeichneten Linien markieren die Gebiete aus denen Teilchen an die Nachbargebiete übertragen werden müssen, da sie in derer Kraftberechnung eine Rolle spielen. Diese Teilchen, die eigentlich nicht zum Nachbargebiet gehören und dort trotzdem für die Berechnung der Interaktionen bekannt sein müssen werden „Geisterteilchen“ genannt. Wie diese Gebiete erkannt und kommuniziert werden, wurde in dieser Arbeit nicht weiter bearbeitet, muss aber für die praktischen Anwendung entwickelt und implementiert werden.

Desweiteren spielt die Betrachtung der Zeitskala eine wichtige Rolle. Ein

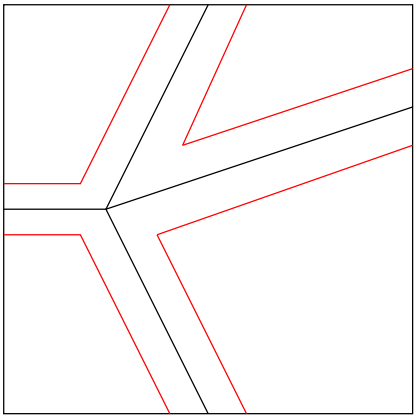


Abbildung 5.1: Zweidimensionales Beispiel für die Gebiete der Geisterpunkte ohne Berücksichtigungen der Randbedingungen

erster sehr naiver Ansatz bestünde darin, zu jedem Zeitschritt, d.h. nach jeder Verschiebung der physikalischen Teilchen, das Loadbalancing Verfahren dieser Arbeit anzuwenden. Diese Vorgehensweise würde sicherstellen, dass zu jedem Zeitpunkt die Arbeitsverteilung ausgewogen wäre. Es wird jedoch schnell klar, dass dieser Ansatz einen großen Overhead zur Folge hätte, da in jedem Zeitschritt so viele Iterationsschritte ausgeführt werden müssten, dass diese nicht mehr in Relation zu der verrichteten Arbeit stehen. Der praktikable Umgang besteht darin, das Loadbalancing-Verfahren nicht in jedem Zeitschritt anzuwenden, sondern nur in bestimmten Abständen. Diese Abstände dürfen nicht zu klein sein, da sonst der Aufwand zu groß ist. Würden sie zu groß gewählt werden, so würde man die gewünschte Lastbalance gar

nicht erst erreichen. Daher wird es noch eine wichtige Aufgabe sein, in der Anwendung zu testen, in welchen Zeitintervallen, das Loadbalancing-Verfahren anzuwenden ist. Auch die Frage, wie viele Iterationsschritte nötig sind um auf Maschinen wie der JUGENE optimal arbeiten zu können muss, unter den betrachteten Gesichtspunkten zum Abbruch des Verfahrens (s. Kapitel 4.1), noch näher eingegangen werden.
Die Resultate dieser Arbeit machen deutlich, dass das „Gradienten basierte Loadbalancing-Verfahren“ die Ergebnisse liefert, um einen guten Arbeitsausgleich schaffen zu können. Diese Erkenntnisse und Ergebnisse bilden somit die Grundlage für die Implementierung in größere Programmpakete, wie etwa MP2C.

Abbildungsverzeichnis

Literaturverzeichnis

[1] Dokumentation des Höchstleistungsrechners JUGENE der Forschungszentrum Jülich GmbH.

[2] Oliver Bücker. *Parallelisierung der Nahfeldberechnung in der schnellen Multipolmethode für stark inhomogene Teilchensysteme*. Forschungszentrum Jülich, Jülich Supercomputer Centre (JSC), Juli 2009. Masterarbeit.

[3] Richard L. Burden. *Numerical Analysis*. Brooks/Cole, 7th edition, 2001.

[4] Florian Fleissner and Peter Eberhard. Parallel load-balanced simulation for short-range interaction particle methods with hierarchical particle grouping based on orthogonal recursive bisection. *International Journal for Numerical Methods in Engineering*, 74:531–553, 2008.

[5] Jens Freche, Wolfgang Frings, and Godehard Sutmann. High throughput parallel-i/o using sionlib for mesoscopic particle dynamics simulations on massively parallel computers. In Barbara Chapman, F. Desprez, G. R. Joubert, A. Lichnewsky, F. J. Peters, and T. Priol, editors, *Parallel Computing: From Multicores and GPU's to Petascale*, volume 19 of *Advances in Parallel Computing*, pages 371 – 378. IOS Press, 2010.

[6] Wolfgang Gürich. Prallele rechnerarchitekturen. Technical report, Forschungszentrum Jülich, Jülich Supercomputer Centre (JSC), 2006.

[7] Rene Halver. *Adaptives Lastbalance-Verfahren für Gebietszerlegung in der Molekulardynamik*. Forschungszentrum Jülich, Jülich Supercomputer Centre (JSC), Februar 2010. Masterarbeit.

[8] Norbert Herrmann. *Höhere Mathematik: Für Ingenieure, Physiker und Mathematiker*. Oldenbourg Wissenschaftsverlag, 2. überarbeitete Auflage 2007. S. 99 - 100.

[9] D. Hilbert. Über eine stetige abbildung einer linie auf ein flächenstück. *Mathematische Annalen*, (38), 1891.

[10] Bernd Körfgen. Prallele algorithmen. Technical report, Forschungszentrum Jülich, Jülich Supercomputer Centre (JSC), 2009.

[11] Carl D. Meyer. *Matrix Analysis and Applied Linear Algebra Book and Solutions Manual*. SIAM: Society for Industrial and Applied Mathematics, Februar 2001. S. 671 - 673.

[12] G.M. Morton. A computer oriented geodetic data base; and a new technique in file sequencing. Technical report, IBM Ltd., 1966.

[13] I. Newton, A.Motte, and J. Machin. *The Principia: Mathematical Principles of Natural Philosophy*. 1789.

[14] G. Sutmann and W. Frings. Extending Scalability of MP^2C to more than 250k Compute Cores. In W. Frings B. Mohr, editor, *Jülich Blue Gene/P Extreme Scaling Workshop 2009*, volume IB-2010-02, 2010.

[15] Godehard Sutmann. Notes on geometrical transformations for plastically deformed domains in particle simulations. not published.

[16] Godehard Sutmann. Molecular dynamics - vision and reality. In J.Grotendorst, D.Marx, and S.Bügel, editors, *Coputational Nanoscience: Do It Yourself!*, pages 159–194. John von Neumann Institute for Computing, Jülich, 2006.

[17] Godehard Sutmann. Computersimulations-methoden in den naturwissenschaften: Klassische molekulardynamik. Technical report, Forschungszentrum Jülich, Jülich Supercomputer Centre (JSC), 2009.

[18] Micheal S. Warren and John K. Salman. A portable parallel particle program. In *Communication*, volume 87, pages 266–290. Mai 1995.

Jül-4330
September 2010
ISSN 0944-2952